Open Geospatial Consortium, Inc.

Date: 2008-09-12

Reference number of this document: OGC 07-138r1

Version: **0.1.0**

Category: Discussion Paper

Editor: Michael Werling

OGC® OWS-5 GeoProcessing Workflow Architecture Engineering Report

Preface

This Engineering Report documents work performed by the OGC Web Services, Phase 5 (OWS-5) initiative of the OGC Interoperability Program. Specifically, this document presents the work completed with respect to the Geoprocessing Workflow (GPW) architecture activities within OWS-5.

This is not a normative document.

Suggested additions, changes, and comments on this draft report are welcome and encouraged. Such suggestions may be submitted by 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.

Forward

Contents

Page

1 Introduction 1
1.1 Scope 1
1.2 Document contributor contact points 1
1.3 Revision history 1
1.4 Future work 2
2 References 2
3 Conventions 3
3.1 Abbreviated terms 3
4 Geoprocessing Workflow Architecture overview 3
5 Web Services Based Architecture 4
5.1 Introduction 4
5.2 Standards 4
5.3 Architecture 5
5.4 Conflation Workflow 6
5.4.1 Infrastructure 7
5.4.2 Design 7
5.4.3 Implementation 9
5.5 Lessons Learned 11
6 RESTful Services Based Architecture 11
6.1 Workflow Engine Interoperability 14
6.2 Security 15
6.3 Workflow Publishing and Discovery 15
6.4 Lessons Learned and Recommendations 17
Annex A XML Documents 19
Bibliography 6

Figures

Page

Figure 1: A high level view of a Web Services based workflow architecture using the Aggregate pattern 5
Figure 2: The infrastructure for the conflation workflow 7
Figure 3: The conflation workflow from the client perspective 8
Figure 4: View of the conflation sequence from inside the process server 9
Figure 5: The conflation workflow in the designer tool 10
Figure 6 Wiring diagram for the NASA EO workflow 12

Figure 7 Enterprise architecture diagram for the EO workflow 13

Tables Page

Table 1: Standards used in the Web Services based workflow architecture 4

OGC® OWS-5 GeoProcessing Workflow Architecture Engineering Report

1 Introduction

1.1 Scope

This OGC® document describes the Workflow Architecture developed in support of Geoprocessing Workflow and Sensor Web Enablement threads of OWS-5. This information includes the overall architecture description, concepts, and issues. It also provides detail on the Conflation Workflow created as an example implementation for geoprocessing in a workflow. This document establishes a sample architecture and associated lessons learned as general guidance.

This OGC™ document is applicable to software system designers, and developers who are creating or using workflows that interact with geospatial web services.

1.2 Document contributor contact points

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

Name	Organization
Michael Werling	Northrop Grumman Corporation – Michael.werling <at> ngc.com
Pat Cappelaere	Vightel Corporation Pat <at> vightel.com

1.3 Revision history

Date	Release	Editor	Primary clauses modified	Description
2007-12-7	0.0.1	Michael Werling	Initial document	Initial Document draft
2008-03-31	0.0.1	Pat Cappelaere	Section 6	Added content for RESTful Services Architecture as used by the SWE thread
2008-05-15	1.0.0	Michael Werling	Document	Edits and final revisions prior to submission
2008-8-1	0.9.0	Carl Reed	Various	Preparation for posting as a DP

1.4 Future work

Additional work should be undertaken to provide guidance to those implementing geoprocessing workflows. Some areas for future work include:

- Web Processing Service (WPS) supports asynchronous processes through the use of a notification page/mechanism, while asynchronous Web Services Description Language (WSDL)/Simple Object Access Protocol (SOAP) based services use a callback mechanism. Additional investigation should be performed to determine best practices when working with asynchronous services and processes.

- Discuss implementing compensation for geospatial services within the workflow. Compensation in terms of workflows involves un-doing actions that have been performed when a latter step in the workflow cannot be completed.

- Discuss invoking OGC services from Business Process Execution Language (BPEL) via Hypertext Transport Protocol (HTTP)/Representational State Transfer (REST) type interfaces rather than SOAP/WSDL. This should be possible for some BPEL engines and would bypass the need to use the SOAP wrapper pattern in some cases, allowing the BPEL engine to directly invoke a WPS.

1.5 Forward

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

2 References

The following documents are referenced in this document. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. For undated references, the latest edition of the normative document referred to applies.

OGC 06-121r3, *OpenGIS® Web Services Common Specification*

OGC 05-007r6, *OpenGIS® Web Processing Service*

OGC 07-160 r1*, OWS-5 Conflation Engineering Report*

OGC 06-187r1, *OWS-4 Workflow IPR*

http://www.w3.org/TR/wsdl, *Web Services Description Language (WSDL) 1.1* http://www.w3.org/TR/soap, *SOAP Version 1.2*

http://www.wfmc.org/standards/docs/WfXML20-200410c.pdf, Workflow Extensible Markup Language (WF-XML)

http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-bpel/ws-bpel.pdf, *Business Process Execution Language for Web Services (BPEL4WS) 1.1*

In addition to this document, this report includes several XML Document listings in Annex A.

3 Conventions

3.1 Abbreviated terms

BPEL	Business Process Execution Language
EO	Earth Observation
GPW	GeoProcessing Workflow
HTTP	Hypertext Transfer Protocol
OGC	Open Geospatial Consortium
OWS	OGC Web Services
REST	Representational State Transfer
SOAP	Simple Object Access Protocol
SWE	Sensor Web Enablement
TQAS	Topological Quality Assessment Service
W3C	World Wide Web Consortium
WfCS	Workflow Chaining Service
WFS	Web Feature Service
WPS	Web Processing Service
WSDL	Web Services Description Language
XML	eXtensible Markup Language

4 Geoprocessing Workflow Architecture overview

The Geo-Processing Workflow (GPW) thread within OGC Web Services 5 (OWS-5) developed and demonstrated interoperability among geospatially-centric processes through service chaining, workflow and web services, with emphasis on the WPS and SOAP bindings. The objective for this thread is described in the OWS-5 RFQ, paragraph 4.2.2-1.a: "Refine and document loosely-coupled integration of OGC Web Services in a service oriented architecture for enterprise. The results will be realized through valuedadded enterprise scenarios that demonstrate the power of interoperability and service-oriented architectures".

The OWS-5 GPW thread integrated and enhanced OGC web services specifications drawing on accomplishments of previous initiatives to meet these objectives. The GPW thread is using data conflation as a scenario to demonstrate service chaining and workflow, web processing services, and service interoperability. The Conflation GPW is implemented using Web Services – Business Process Execution Language (WS-BPEL) to chain together multiple Web Services that expose their interfaces using WSDL and exchange messages using SOAP. The Conflation GPW builds on work done to create the Topological Quality Assessment Service (TQAS) workflow in OWS-4.

The SWE thread is also building services based workflows. This report describes the REST base architecture implemented by the SWE thread using an Earth Observation (EO) use case. REST is an alternate architecture approach to the Web Services architecture used by the GPW thread.

5 Web Services Based Architecture

5.1 Introduction

There are many approaches to implementing an architecture that uses Service Oriented Architecture (SOA) concepts. Use of Web Services is one of the most frequently referred to ways to implement SOA, although other approaches, notably RESTful Oriented Architecture described below, are also used to implement a services based enterprise. A Web Services based architecture uses a set of open specifications to describe the interactions between components of the architecture. By following these standards the Web Services achieve interoperability independent of the underlying implementation of the individual services.

5.2 Standards

An architecture based on Web Services uses many different standards to define the interaction between different components of the system. The standards listed in Table 1

Table 1: Standards used in the Web Services based workflow architecture

Standard	Version	Specification Body	Use in GPW Thread
Business Process Execuation Language for Web Services (BPEL4WS)	BPEL4WS 1.1	N/A	Scripts run by the workflow engine follow this version of the standard*
Web Feature Service	1.1.0	OGC	Data sources and results for conflation are published using WFS
Web Processing Service (WPS)	1.1	OGC	WPS wrapper to workflow called by integrated client
Web Services Description	1.1	W3C	Web Services (e.g. conflation, TQAS),

Language (WSDL)			business processes
Simple Object Access Protocol	1.2	W3C	
Web Feature Service	1.1.0	OGC	Publishing of input data for conflation
HTTP		W3C	Interaction between services, client, and the WPS wrapper is through HTTP

*Note that BPEL4WS is now called WS-BPEL 2.0, a standard managed by Organization for the Advancement of Structured Information Standards (OASIS).

The main OGC standard used in the Conflation Workflow use case for OWS-5 was WPS. WPS defines a standardized interface for publishing geospatial processes. In the context of service chaining and workflow the services are generally used to perform actions, as opposed to data retrieval and visualization; WPS is suited to this pattern of interaction. Services based on WFS were also used as data sources passed into the processing services.

5.3 Architecture

The GPW architecture for OWS-5 used an Aggregate service, or opaque, pattern. This pattern is described in ISO 19119. In the implementation of this workflow architecture the client is only aware of the service published by the process server. The client does not know, or need to know, that the module they are invoking is made up of a composition of other services. Within the process one or more other services are invoked. In the architecture implemented for the GPW thread these services are not aware of each other; all services work in a query-response sequence with the process server. Figure 1 shows a high-level view of the components needed in a workflow architecture.

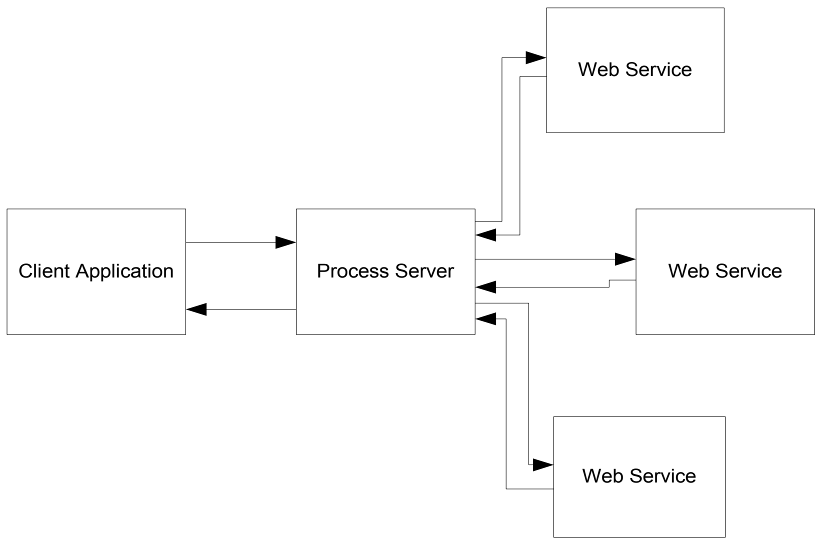


Figure 1: A high level view of a Web Services based workflow architecture using the Aggregate pattern

The process server is the central actor in the architecture. In the Web Services based architecture the process server executes processes designed using BPEL. BPEL requires that all services publish their capabilities using WSDL and interact using SOAP messages. Therefore any services using WPS (or other OGC standards) must implement the WSDL and SOAP bindings for that specification.

Generally BPEL processes are presented to the world using WSDL and SOAP. In this way processes appear as regular Web Services to any client or other system that wants to invoke it. Through this mechanism processes can be aggregated and chained together to build more specific implementations. For the purposes of the OWS-5 GPW thread the process is also exposed using a WPS interface. This is done following the Adapter Pattern as described in the OWS-4 Workflow Interoperability Project Report (IPR). Clients that already interact with WPS based services can easily access the workflow through the WPS interface.

5.4 Conflation Workflow

The GPW thread built a workflow based on data conflation as a use case to construct the architecture and demonstrate the concepts discussed in this report. The Conflation Engineering Report, OGC 07-160 r1, describes the concept, implementation, and issues specific to the task of conflation. At the level of this architecture report, conflation is regarded as a service that combines two data sources based on some set of rules and provides a combined result data set. Conflation is potentially a time consuming process, depending on the size and complexity of the input data sets. The workflow must be able to handle potentially long-running, or asynchronous, services.

This workflow followed a bottom-up approach to the design. A bottom-up approach uses web services that have already been defined and chains them together in a workflow. The opposite, a top-down approach, creates the workflow and then specifies the requirements for the interfaces and implementation of the services. Given the collaborative and distributed nature of the GPW team for OWS-5 the bottom-up approach gave the most flexibility to developing the individual services.

5.4.1 Infrastructure

For OWS-5 the GPW thread used a distributed environment consisting of systems hosted by different organizations at geographically distributed sites. This infrastructure demonstrates the flexibility of a services based enterprise to incorporate applications in a network-centric environment. Figure 2 shows the servers and systems involved in the implemented workflow for this OWS-5 thread.

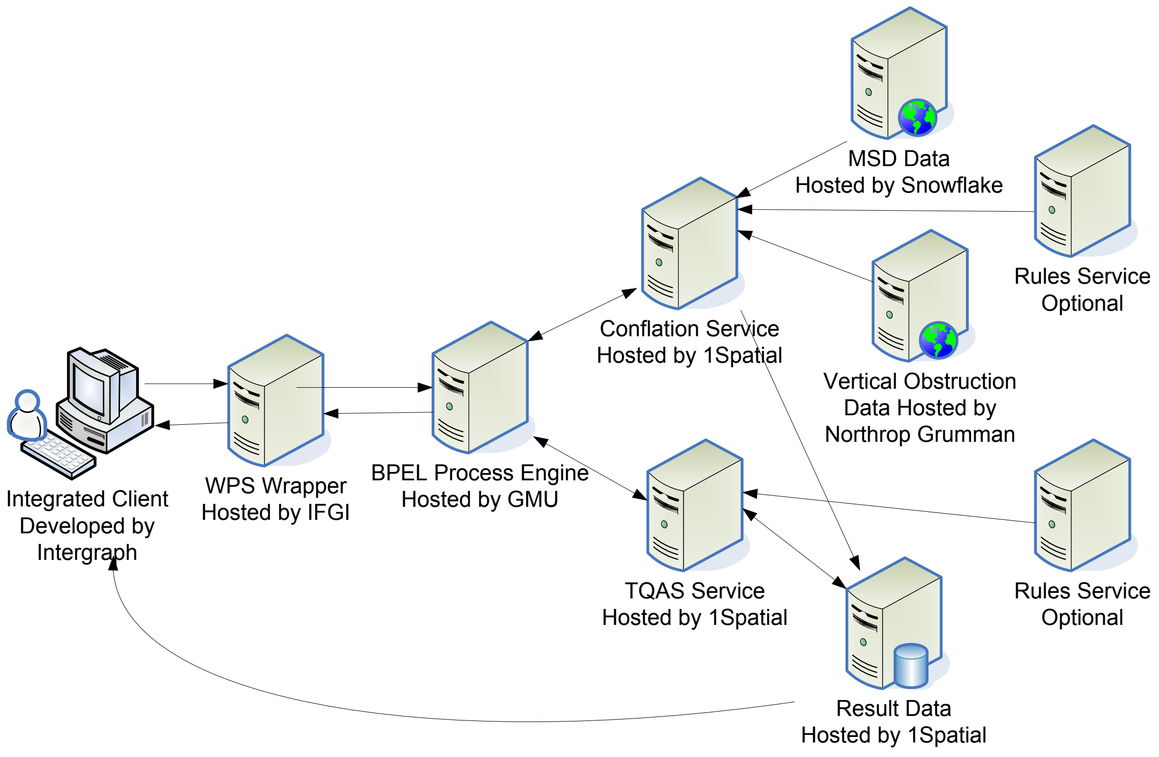


Figure 2: The infrastructure for the conflation workflow

5.4.2 Design

The design for the conflation workflow was created based on the sequence of operations and information interchanges to needed to execute the services. From the client perspective the workflow is a WPS that is invoked and provides an asynchronous response in accordance with the WPS specification. The resulting data is passed to the client as the address of the WFS publishing the data. Figure 3 shows the sequence of activities as the WPS is invoked, which in turn, invokes the BPEL process residing on the process server.

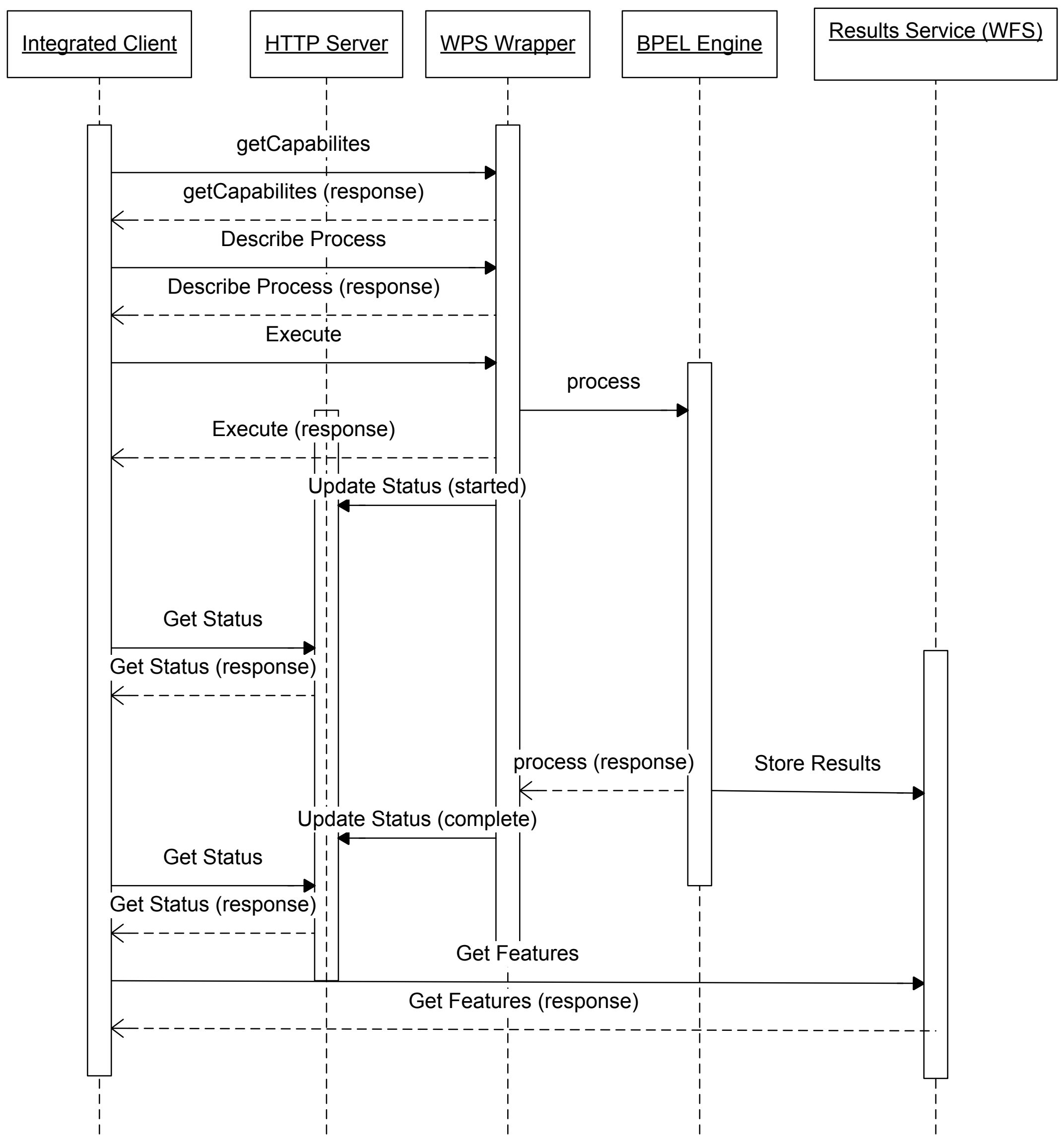


Figure 3: The conflation workflow from the client perspective

Figure 4 shows the sequence of activities that take place within the process server as the BPEL script is executed.

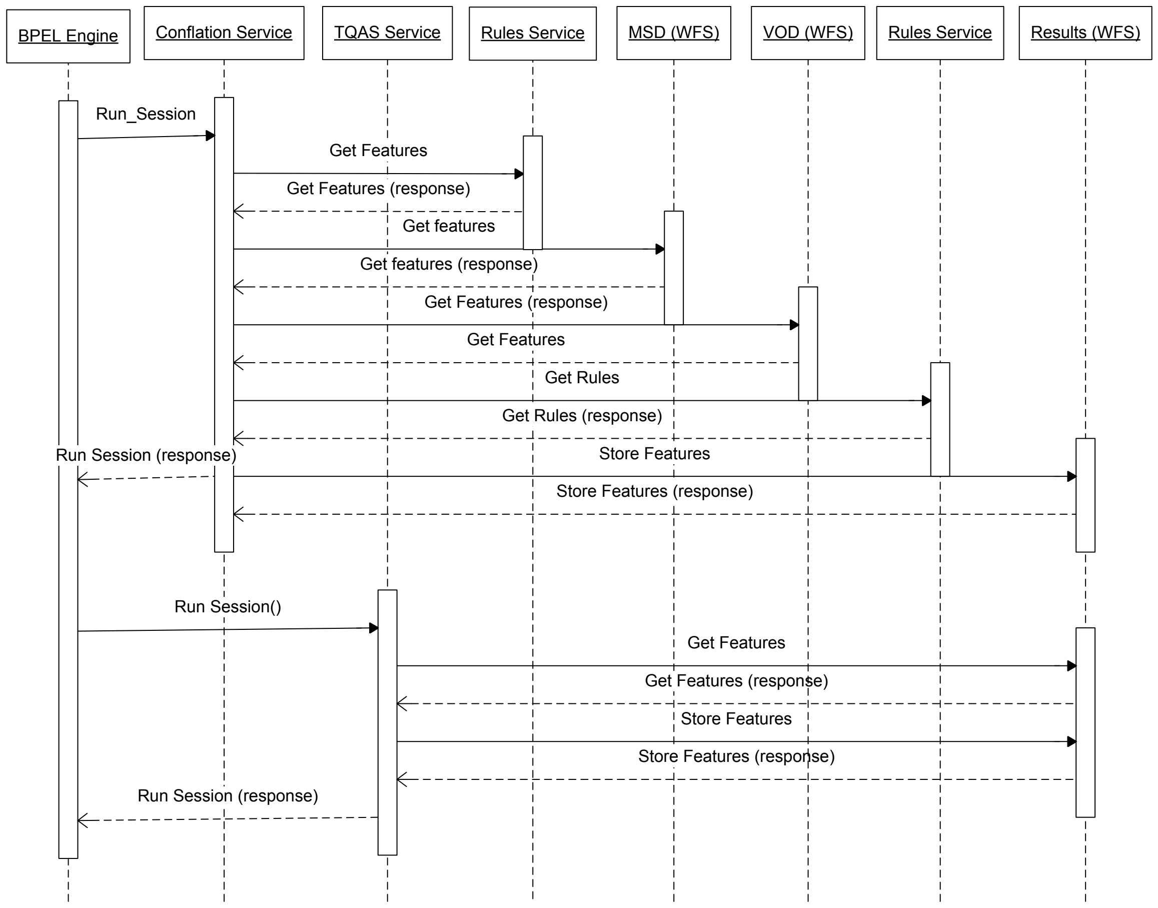


Figure 4: View of the conflation sequence from inside the process server

5.4.3 Implementation

The conflation workflow was developed using Oracle BPEL Designer 10g (10.1.2) and deployed to the GMU BPEL Power process engine. Figure 5 shows a graphic of the BPEL script in the designer tool. Most tools provide this type of graphic interface to construct the workflow. The resulting BPEL script is listed in Annex A.

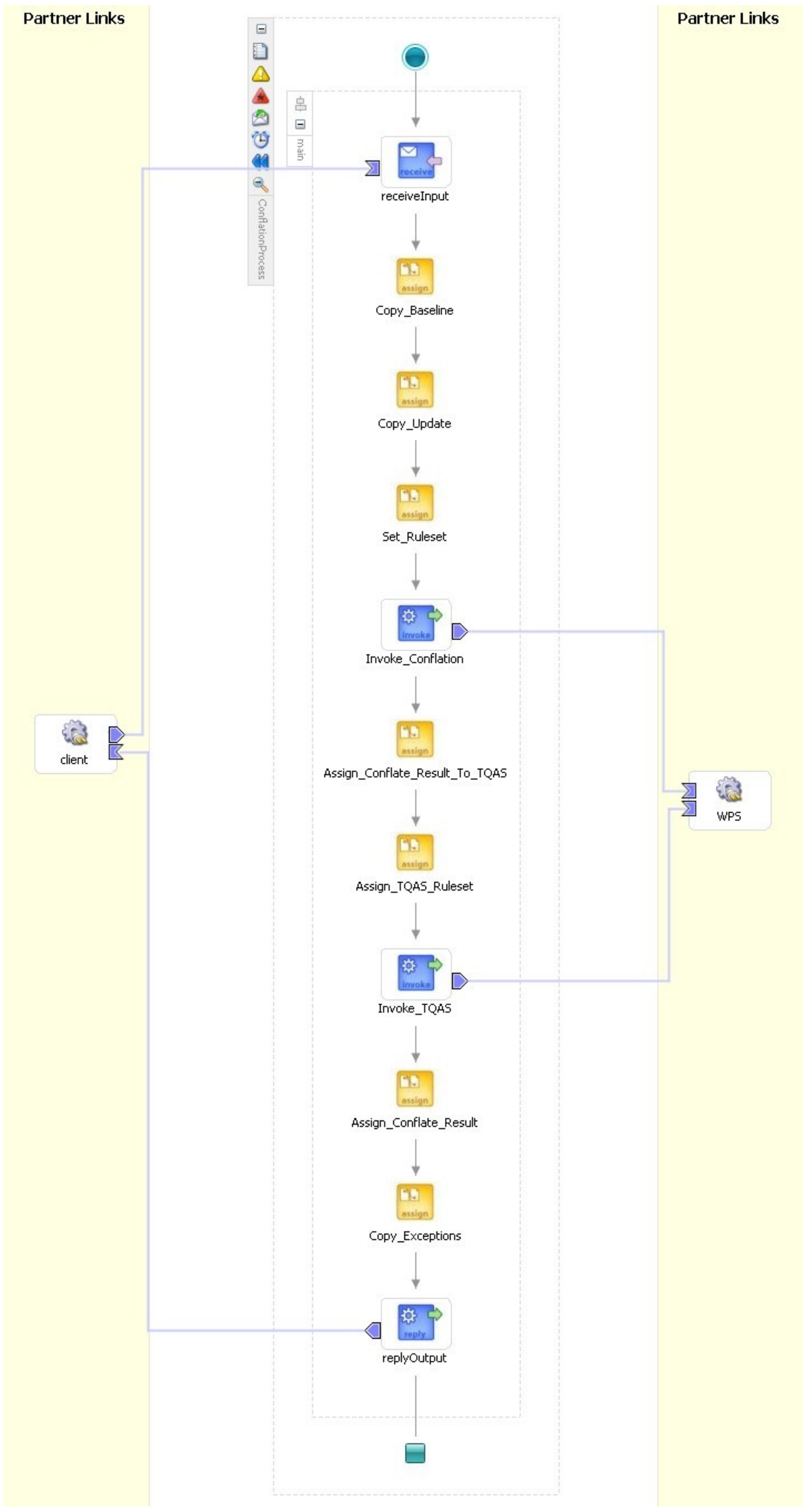


Figure 5: The conflation workflow in the designer tool

5.5 Lessons Learned

This exercise explored the use of OGC standards within a Web Services based process. One area for further experiments is the interaction of asynchronous services and/or processes between the two different approaches (Web Services and WPS). The use of WPS services within a workflow is possible, provided the WPS has SOAP bindings and a published WSDL. However, the status and success reporting mechanism for long-running or asynchronous services is via a published web address. This web address must be polled during the execution of the process so until the service succeeds or fails. WSDL/SOAP based asynchronous services generally respond with status via a callback function, letting the process set up a receive operation or an alert signal. There is currently no clear way to bridge these two technologies; some update to the WPS specification may be required to accommodate this functionality.

Within the business process created for this exercise there was little parameter validation and error checking. Additional error checking could provide the opportunity to explore compensation within the process. Compensation is a mechanism to roll back or undo changes in early steps of a process when a later step encounters a failure state. Integrating this with geospatial services would explore how to undo changes, integrated with the compensation mechanism of the process server.

A fully engineered version of a GPW implementation would have to account for many different factors, including security, load balancing, and handling simultaneous user requests. The implementation in this exercise was not intended to meet all requirements of a fielded system, but to demonstrate the interaction of standards based geospatial services in a workflow.

6 RESTful Services Based Architecture

The EO/NRE scenario for OWS-5 was focused on demonstrating an Enterprise Architecture based on RESTFul services.

This effort was focused into three areas:

- Workflow Engine Interoperability
- Security
- Workflow Publishing and Discovery

Many organizations participated in the interoperability demonstration as part of the OWS-5 effort including Vightel, NASA, JPL, Northrop Gruman IT, George Mason University, UAH or as part of their on-going efforts to support the NASA SensorWeb funded by NASA Earth Science Technology Office (West Virginia High Tech Foundation). The wiring diagram in Figure 6 shows the services and participants that interacted to create this workflow.

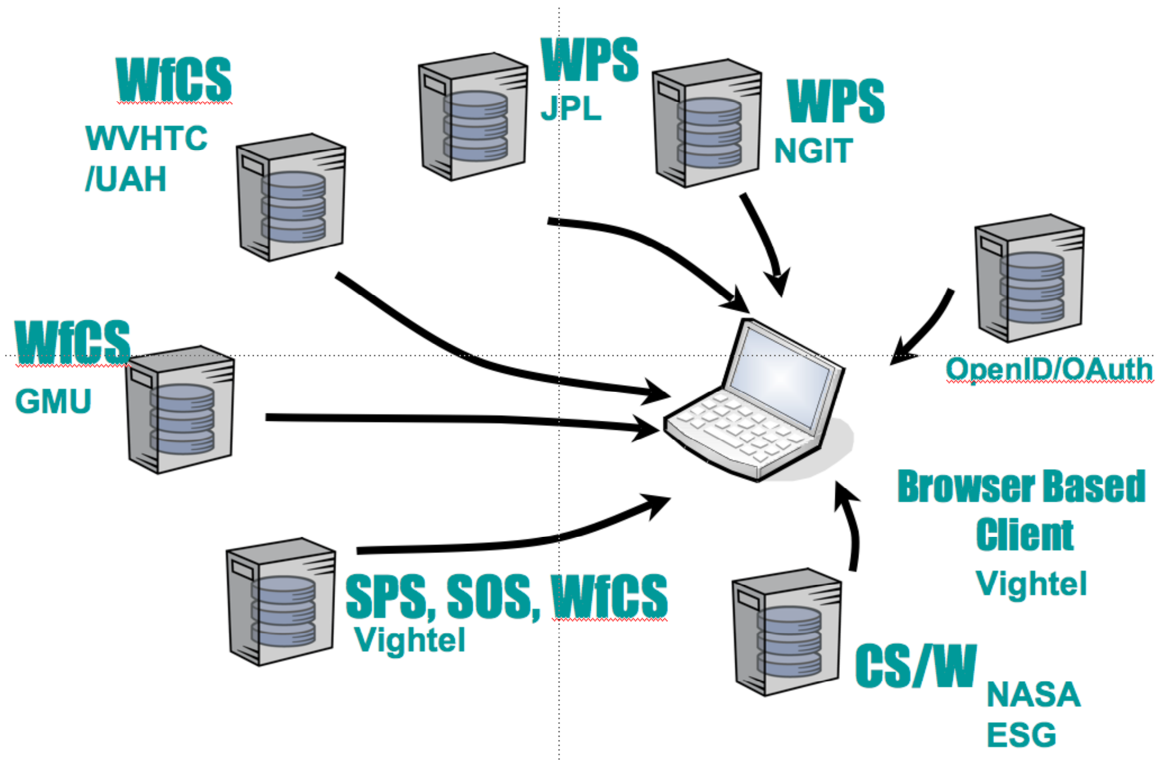


Figure 6 Wiring diagram for the NASA EO workflow

The actual instantiation of the wiring diagram is shown in Figure 7.

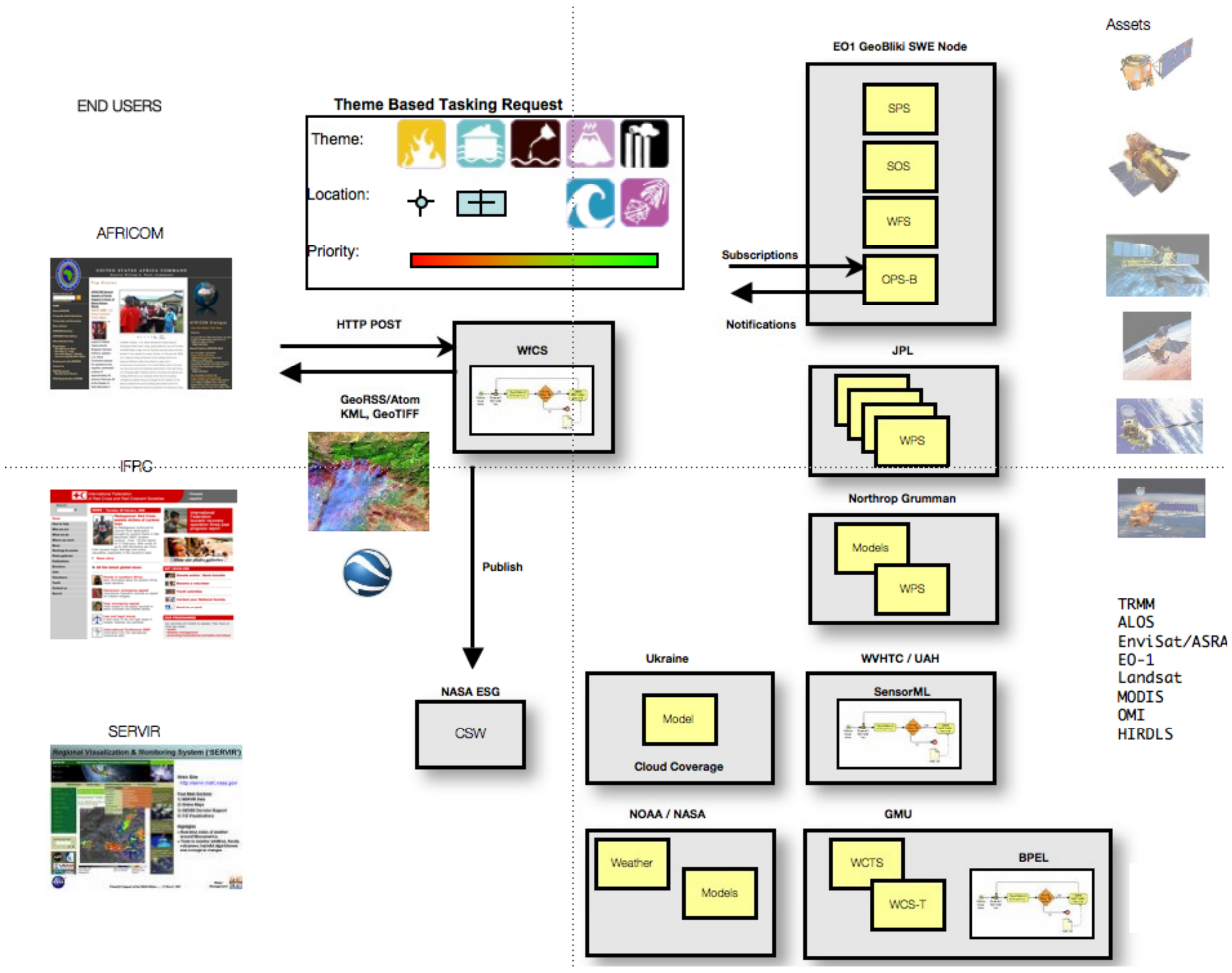


Figure 7 Enterprise architecture diagram for the EO workflow

This architecture is tailored to support end users from various communities such as SERVIR, International Federation of the Red Cross or even DOD organizations such as USAFRICOM. Those end-users interface with the system via workflows that allow them to specify locations and themes of interest. The workflows orchestrate many OGC web services. Raw data is acquired from various assets such as EO-1 and processed by many different web processing services. Relevant data are then published and distributed to the end-users in a popular format such as GeoAtom and KML.

6.1 Workflow Engine Interoperability

Several workflow engines were used to support the EO/NRE scenario. GMU was providing the GeoBrain BPEL engine, the West Virginia High Tech Foundation was developing a workflow engine based on the UAH Space ToolKit and Vightel was providing GeoBPMS based on the OpenWFE Ruby engine. This work was funded by the OGC OWS-5 testbed and ESTO.

Interoperability of the workflow engines was demonstrated following the development of an interface specification for Workflow Chaining Services (WfCS) based on the WfMC Wf-XML 2.0 specification, available at http://www.wfmc.org/standards/docs/WfXML20-200410c.pdf. A draft of the RESTFul specification is available at: http://groups.google.com/group/wfxml. This allowed a seamless and consistent interface between the workflow engines. A workflow participant at the highest level was able to trigger lower level workflows in SensorML and BPEL for specific processing of the data based on theme selection, as shown in Figure 8.

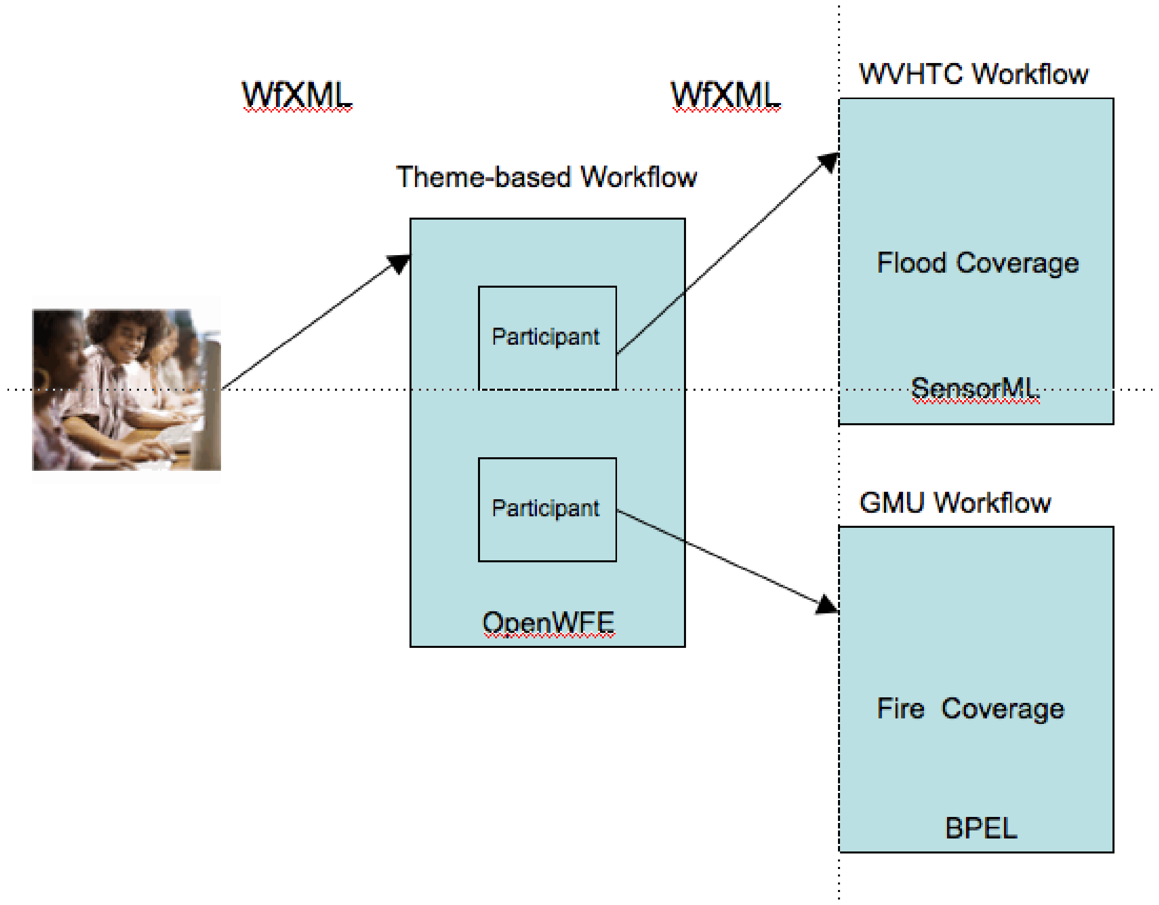


Figure 8 Interaction between clients and workflows using WfXML

6.2 Security

Security is a big issue. We need to authorize and authenticate users coming in from various communities without having to manage their credentials (username/passwords) and we need to have secured transactions between web services. The current basic authentication approach used in OWS-4 is not acceptable as it amount to passing a username and password in the clear within the transaction.

User management is an issue that needed to be addressed. Several communities of interests have already been identified to access the NASA SensorWeb including the Science Community, the International Federation of the Red Cross, Police, Fire Fighters, and even several DOD agencies such as DIA and NRO.

To be scalable, Identity Management has to be distributed and managed by the respective communities. A trust relationship has to be established within the federation. This trust relationship was simply assumed for the purpose of the demonstration and every identity provider was accepted. This will not be the case for an operational system.

OpenID 2.0 was selected to support those requirements and the JanRain Ruby Open Source libraries were used. Our application was modified to allow users to present their openid credentials as an alternative to their usernames and passwords. The system automatically redirected those users to their Open Identity Providers for Authentication. The Profile Exchange was used to get access the various user attributes and automatically create an account for that user if necessary.

Another issue to tackle was the delegation of user authority to workflows. A workflow engine is a web service consumer that needs to have privilege access to other services in order to act on behalf of the user. OAuth 1.0 was selected to support this requirement. Another simplification of this protocol was made by allowing users to pre-grant access to specific security realms. In that particular case, a transaction can be secured using the HTTP authentication header and the user openid can be passed within one of the parameters. A digital signature is computed using a pre-agreed upon secret between the consumer and provider. This signature is generated using several oauth attributes such as nonce, timestamp… and the message itself. The provider can recomputed the signature and compare it with what was provided by the consumer.

The provider used the openid to determine the access granted by the identified user for that consumer by checking a grants table. This allowed delegation of user authority to the workflow.

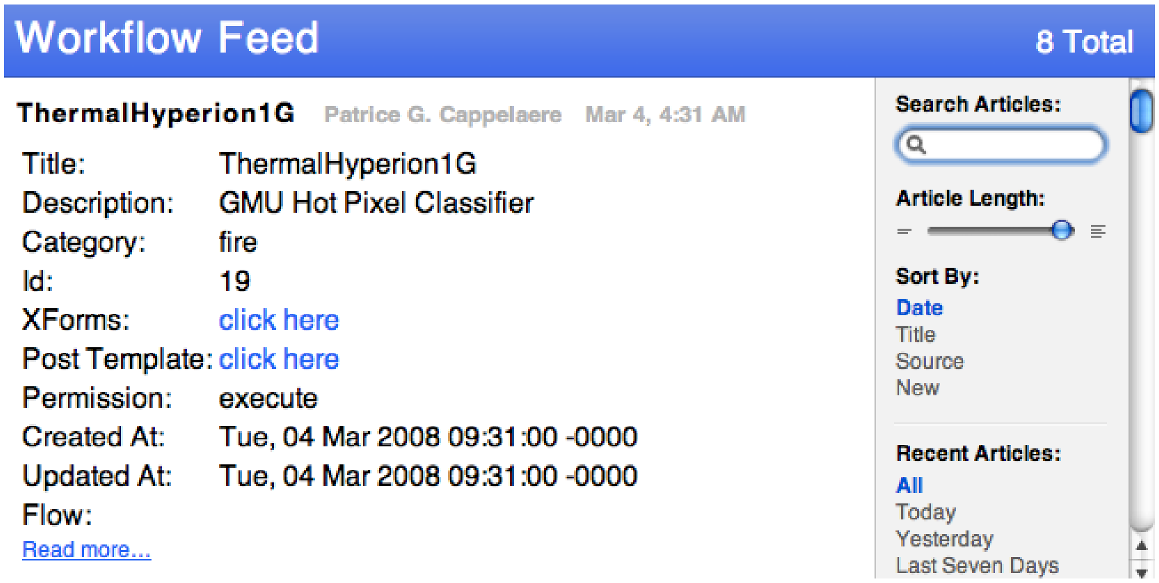
6.3 Workflow Publishing and Discovery

WfXML-R relies heavily on the Atom Publishing Protocol or APP. This protocol was demonstrated as a way to publish workflows that can be discovered by the user community using standard tools such as feed readers.

A user can access the APP service document and discover the available collections of the Workflow Chaining Service. An Atom feed is generated when a user accesses the collection. The listing below shows the XML for an entry in the Atom feed, taken from http://geobpms.geobliki.com/wfcs/workflows.atom.

```
<?xml version="1.0" encoding="UTF-8"?>
<feed xmlns="http://www.w3.org/2005/Atom">
  <title>Workflow Feed</title>
  <link href="http://geobpms.geobliki.com/wfcs/workflows.atom"
rel="self"/>
  <link
href="http://geobpms.geobliki.com/wfcs/workflows.html?type=text%2
Fhtml" rel="alternate"/>
  <link href="http://geobpms.geobliki.com/workflow_search.xml"
title="Content Search" rel="search"
type="application/opensearchdescription+xml"/>
  <id>http://geobpms.geobliki.com/wfcs/workflows</id>
  <updated>2008-05-14T20:19:08Z</updated>
  <author>
    <name>GeoBPMS</name>
  </author>
  <entry xmlns:geobpms="http://geobpms.geobliki.com/1.0">
    <title>ThermalHyperion1G</title>
    <link
href="http://geobpms.geobliki.com/workflows/show/19.html"
type="text/html"/>
    <link href="http://localhost:4000/wfcs/workflows/19.atom"
rel="alternate" type="application/atom+xml"/>
    <link href="http://localhost:4000/wfcs/workflows/19.atom"
rel="edit" type="application/atom+xml"/>
    <id>http://geobpms.geobliki.com/wfcs/workflows/19</id>
    <category scheme="http://geobpms/1.0" term="fire"/>
    <updated>2008-03-04T09:31:00Z</updated>
    <author>
      <name>Patrice G. Cappelaere</name>
      <email>pat@cappelaere.com</email>
      <uri></uri>
    </author>
    <summary>GMU Hot Pixel Classifier</summary>
    <content type="html">
<![CDATA[<table><tr><td>Title:</td><td>ThermalHyperion1G</td></tr
><tr><td>Description:</td><td>GMU Hot Pixel
Classifier</td></tr><tr><td>Category:</td><td>fire</td></tr><tr><
td>Id:</td><td>19</td></tr><tr><td>XForms:</td><td><a
href='http://geobpms.geobliki.com/launch/launch/19'>click
here</a></td></tr><tr><td>Post Template:</td><td><a
href='http://geobpms.geobliki.com/launch/template/19.xml'>click
here</a></td></tr><tr><td>Permission:</td><td>execute</td></tr><t
r><td>Created At:</td><td>Tue, 04 Mar 2008 09:31:00 -
0000</td></tr><tr><td>Updated At:</td><td>Tue, 04 Mar 2008
09:31:00 -
0000</td></tr><tr><td>Flow:</td><td></td></tr></table>]]>
</content>
    <geobpms:item_type>workflows</geobpms:item_type>
    <geobpms:permission>execute</geobpms:permission>
  </entry>
```

This information is rendered in a viewer application, shown in Figure 9.



Workflow Feed 8 Total

ThermalHyperion1G Patrice G. Cappelaere Mar 4, 4:31 AM

Title:	ThermalHyperion1G
Description:	GMU Hot Pixel Classifier
Category:	fire
Id:	19
XForms:	click here
Post Template:	click here
Permission:	execute
Created At:	Tue, 04 Mar 2008 09:31:00 -0000
Updated At:	Tue, 04 Mar 2008 09:31:00 -0000
Flow:	

Read more...

Figure 9 The Atom feed in the viewing application

For the purpose of the demonstration, the feed was published using Google Feedburner (http://www.feedburner.com/fb/a/home). This made it accessible using Google Reader (as well as many other feed aggregators/readers).

The feed was also entered as a custom entry in the NASA Earth Science Gateway (ESG) Portal OGC Catalog.

6.4 Lessons Learned and Recommendations

Many workflow engines can be used and made inter-operable using WfXML.

Asynchronous operation support is critical. This is built-in within a workflow engine. Asynchronous access within the WPS needs to be explored.

A RESTFul security model based on OpenID and OAuth for OGC services should be further explored.

From a user perspective, there are no differences between an SPS, WPS and a Workflow. As we are [re]defining an interoperable and secure API, we need to consider the harmonization of these standards. The relationship (and possible harmonization) of SPS and WPS with other workflow engines, such as WfXML, needs to be explored further. This would help the user community in providing a consistent API across many OGC similar services.

Catalogs have been an issue. They can be complex, hard to use, difficult to manage and to keep up-to-date.

A Web 2.0 approach based on the Atom Publishing Protocol provides a simple alternative that can be used for publication and discovery. More work needs to be done in that area to make sure that enough information is provided in the atom entries. However, it appears that a feed aggregator coupled with an OpenSearch front-end could very well be a solution.

Annex A

XML Documents

A.1 Conflation BPEL

```
<?xml version = "1.0" encoding = "UTF-8" ?>
<!--
/////////////////////////////////////////////////////////////////
  Oracle JDeveloper BPEL Designer

  Created: Tue Feb 26 09:26:44 EST 2008
  Author:  werlimi
  Purpose: Synchronous BPEL Process
///////////////////////////////////////////////////////////////-->
<process name="ConflationProcess"
targetNamespace="http://xmlns.ows5.org/ConflationProcess"
xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:xp20="http://www.oracle.com/XSL/Transform/java/oracle.tip.p
c.services.functions.Xpath20"
xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-
process/" xmlns:ns1="http://invokews.ows5.onespatial.com/"
xmlns:wps_conflate="http://invokews.ows5.onespatial.com/"
xmlns:ldap="http://schemas.oracle.com/xpath/extension/ldap"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:client="http://xmlns.ows5.org/ConflationProcess"
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.p
c.services.functions.ExtFunc">

<!--
////////////////////////////////////////////////////////////////
      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:ConflationProcess"
myRole="ConflationProcessProvider"/>
    <partnerLink name="WPS" partnerLinkType="ns1:WPSInvokeWS_PL"
myRole="WPSInvokeWS_Role" partnerRole="WPSInvokeWS_Role"/>
```

```
  </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:ConflationProcessRequestMessage"/><!--
Reference to the message that will be returned to the requester--
>
    <variable name="outputVariable"
messageType="client:ConflationProcessResponseMessage"/>
    <variable name="Conflate_InputVariable"
messageType="ns1:ExecuteProcess_Conflate"/>
    <variable name="Conflate_OutputVariable"
messageType="ns1:ExecuteProcess_ConflateResponse"/>
    <variable name="TQAS_OutputVariable"
messageType="ns1:ExecuteProcess_TQASResponse"/>
    <variable name="TQAS_InputVariable"
messageType="ns1:ExecuteProcess_TQAS"/>
  </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 ConflationProcess.wsdl) -->
    <receive name="receiveInput" partnerLink="client"
portType="client:ConflationProcess" operation="conflation"
variable="inputVariable" createInstance="yes"/><!-- Generate
reply to synchronous request -->
    <assign name="Copy_Baseline">
      <copy>
        <from variable="inputVariable" part="payload"
query="/client:ConflationProcessProcessRequest/client:WFS_Baselin
e_Address"/>
        <to variable="Conflate_InputVariable" part="parameters"
query="/ns1:conflate/ns1:baseline"/>
      </copy>
    </assign>
    <assign name="Copy_Update">
      <copy>
        <from variable="inputVariable" part="payload"
query="/client:ConflationProcessProcessRequest/client:WFS_Update_
Address"/>
```

```
        <to variable="Conflate_InputVariable" part="parameters"
query="/ns1:conflate/ns1:update"/>
      </copy>
    </assign>
    <assign name="Set_Ruleset">
      <copy>
        <from expression="'fast-test'"/>
        <to variable="Conflate_InputVariable" part="parameters"
query="/ns1:conflate/ns1:ruleset"/>
      </copy>
    </assign>
    <invoke name="Invoke_Conflation" partnerLink="WPS"
portType="ns1:WPSInvokeWS" operation="ExecuteProcess_Conflate"
inputVariable="Conflate_InputVariable"
outputVariable="Conflate_OutputVariable"/>
    <assign name="Assign_Conflate_Result_To_TQAS">
      <copy>
        <from variable="Conflate_OutputVariable"
part="parameters" query="/ns1:conflateResponse/ns1:result"/>
        <to variable="TQAS_InputVariable" part="parameters"
query="/ns1:TQAS/ns1:dataset"/>
      </copy>
    </assign>
    <assign name="Assign_TQAS_Ruleset">
      <copy>
        <from expression="'fast-test'"/>
        <to variable="TQAS_InputVariable" part="parameters"
query="/ns1:TQAS/ns1:ruleset"/>
      </copy>
    </assign>
    <invoke name="Invoke_TQAS" partnerLink="WPS"
portType="ns1:WPSInvokeWS" operation="ExecuteProcess_TQAS"
inputVariable="TQAS_InputVariable"
outputVariable="TQAS_OutputVariable"/>
    <assign name="Assign_Conflate_Result">
      <copy>
        <from variable="Conflate_OutputVariable"
part="parameters" query="/ns1:conflateResponse/ns1:result"/>
        <to variable="outputVariable" part="payload"
query="/client:ConflationProcessProcessResponse/client:Result_WFS
_Address"/>
      </copy>
    </assign>
    <assign name="Copy_Exceptions">
      <copy>
        <from variable="Conflate_OutputVariable"
part="parameters" query="/ns1:conflateResponse/ns1:exceptions"/>
        <to variable="outputVariable" part="payload"
query="/client:ConflationProcessProcessResponse/client:Exceptions
"/>
      </copy>
    </assign>
    <reply name="replyOutput" partnerLink="client"
portType="client:ConflationProcess" operation="conflation"
variable="outputVariable"/>
  </sequence>
</process>
```

A.2 Conflation Process WSDL

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="ConflationProcess"

targetNamespace="http://xmlns.ows5.org/ConflationProcess"
             xmlns="http://schemas.xmlsoap.org/wsdl/"

xmlns:client="http://xmlns.ows5.org/ConflationProcess"

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.ows5.org/ConflationProcess"
          xmlns="http://www.w3.org/2001/XMLSchema">
   <element name="ConflationProcessProcessRequest">
    <complexType>
     <sequence>
      <!-- First element is the baseline.  For OWS-5 this must be
            the MSD using GML3 -->
      <element name="WFS_Baseline_Address" type="string"/>
      <!-- Second element is the update dataset.  For OWS-5 this
must be the VO using GML2 -->
      <element name="WFS_Update_Address" type="string"/>
      <element name="Rules_Source_Address" type="string"/>
     </sequence>
    </complexType>
   </element>
   <element name="ConflationProcessProcessResponse">
    <complexType>
     <sequence>
      <element name="Result_WFS_Address" type="string"/>
      <element name="Exceptions" 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="ConflationProcessRequestMessage">
  <part name="payload"
element="client:ConflationProcessProcessRequest"/>
 </message>
 <message name="ConflationProcessResponseMessage">
  <part name="payload"
element="client:ConflationProcessProcessResponse"/>
 </message>
 <!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   PORT TYPE DEFINITION - A port type groups a set of operations
into a logical service unit.
   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
 <!-- portType implemented by the ConflationProcess BPEL process
-->
 <portType name="ConflationProcess">
  <operation name="conflation">
   <input message="client:ConflationProcessRequestMessage"/>
   <output message="client:ConflationProcessResponseMessage"/>
  </operation>
 </portType>
 <!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   PARTNER LINK TYPE DEFINITION
   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
 <plnk:partnerLinkType name="ConflationProcess">
  <plnk:role name="ConflationProcessProvider">
   <plnk:portType name="client:ConflationProcess"/>
  </plnk:role>
 </plnk:partnerLinkType>
</definitions>
```

A.4 1Spatial Studio Service WSDL

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"

xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
             xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
             xmlns:xsd="http://www.w3.org/2001/XMLSchema"
             xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
             xmlns:tns="http://invokews.ows5.onespatial.com/"

xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/"
             name="WPS"

targetNamespace="http://invokews.ows5.onespatial.com/">
    <types>
        <schema xmlns="http://www.w3.org/2001/XMLSchema"
                xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
                xmlns:soap11-
enc="http://schemas.xmlsoap.org/soap/encoding/"

targetNamespace="http://invokews.ows5.onespatial.com/"
                elementFormDefault="qualified">

            <element name="conflate"
type="tns:conflateParameters"/>
            <complexType name="conflateParameters">
                <sequence>
                    <element name="baseline" type="string"
nillable="false"/>
                    <element name="update" type="string"
nillable="false"/>
                    <element name="ruleset" type="string"
nillable="false"/>
                </sequence>
            </complexType>

            <element name="conflateResponse"
type="tns:actualConflateResponse"/>
            <complexType name="actualConflateResponse">
                <sequence>
                    <element name="status" type="string"
nillable="true"/>
                    <element name="result" type="string"
nillable="true"/>
                    <element name="exceptions" type="string"
nillable="true"/>
                </sequence>
            </complexType>

            <element name="TQAS" type="tns:TQASParameters"/>
            <complexType name="TQASParameters">
                <sequence>
                    <element name="dataset" type="string"
nillable="false"/>
                    <element name="ruleset" type="string"
nillable="false"/>
                </sequence>
            </complexType>
            <element name="TQASResponse"
type="tns:actualTQASResponse"/>
            <complexType name="actualTQASResponse">
                <sequence>
                    <element name="status" type="string"
nillable="true"/>
                    <element name="result" type="string"
nillable="true"/>
                    <element name="exceptions" type="string"
nillable="true"/>
                </sequence>
            </complexType>


      <element name="GetCapabilities" >
      <complexType  >
         <sequence />
      </complexType>
      </element>

      <element name="GetCapabilitiesResponse"
type="tns:actualGetCapabilitiesResponse"/>

            <complexType name="actualGetCapabilitiesResponse">
                <sequence>
                    <element name="return" type="string"
nillable="true"/>
                </sequence>
            </complexType>

      <element name="DescribeProcess" >
      <complexType  >
         <sequence />
      </complexType>
      </element>

      <element name="DescribeProcessResponse"
type="tns:actualDescribeProcessResponse"/>
            <complexType name="actualDescribeProcessResponse">
                <sequence>
```

```
                    <element name="return" type="string"
nillable="true"/>
                </sequence>
            </complexType>
        </schema>
    </types>

    <message name="GetCapabilities">
        <part name="parameters" element="tns:GetCapabilities"/>
    </message>
    <message name="GetCapabilitiesResponse">
        <part name="parameters"
element="tns:GetCapabilitiesResponse" />
    </message>

    <message name="DescribeProcess">
        <part name="parameters" element="tns:DescribeProcess"/>
    </message>
    <message name="DescribeProcessResponse">
        <part name="parameters"
element="tns:DescribeProcessResponse" />
    </message>

    <message name="ExecuteProcess_Conflate">
        <part name="parameters" element="tns:conflate"/>
    </message>
    <message name="ExecuteProcess_ConflateResponse">
        <part name="parameters" element="tns:conflateResponse" />
    </message>

    <message name="ExecuteProcess_TQAS">
        <part name="parameters" element="tns:TQAS"/>
    </message>
    <message name="ExecuteProcess_TQASResponse">
        <part name="parameters" element="tns:TQASResponse" />
    </message>

    <portType name="WPSInvokeWS">

      <operation name="GetCapabilities">
      <documentation>Get Capabilities request for the WPS
Service. Please consult the OGC Web Site for the WPS 1.0.0
specification.</documentation>
            <input message="tns:GetCapabilities"/>
            <output message="tns:GetCapabilitiesResponse"/>
     </operation>

      <operation name="DescribeProcess">
      <documentation>Describe Process request for the WPS
Service. Please consult the OGC Web Site for the WPS 1.0.0
specification.</documentation>
            <input message="tns:DescribeProcess"/>
            <output message="tns:DescribeProcessResponse"/>
     </operation>

     <operation name="ExecuteProcess_Conflate">
      <documentation>WPS Conflation process.</documentation>
      <input message="tns:ExecuteProcess_Conflate" />
      <output message="tns:ExecuteProcess_ConflateResponse" />
     </operation>

     <operation name="ExecuteProcess_TQAS">
      <documentation>TQAS process.</documentation>
      <input message="tns:ExecuteProcess_TQAS" />
      <output message="tns:ExecuteProcess_TQASResponse" />
     </operation>
    </portType>

    <binding name="WPSInvokeWSSoapHttp" type="tns:WPSInvokeWS">
        <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation name="GetCapabilities">
            <soap:operation soapAction=""/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
        </operation>
        <operation name="DescribeProcess">
            <soap:operation soapAction=""/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
       </operation>
        <operation name="ExecuteProcess_Conflate">
            <soap:operation soapAction=""/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
        </operation>

        <operation name="ExecuteProcess_TQAS">
            <soap:operation soapAction=""/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
```

```
                <soap:body use="literal"/>
            </output>
        </operation>
    </binding>

    <service name="WPS">
       <port name="WPSInvokeBean"
binding="tns:WPSInvokeWSSoapHttp">
           <soap:address
location="http://ogcows.1spatial.com:8082/axis2/services/WPS"/>
        </port>
    </service>

  </definitions>
```

Bibliography

[1] Guidelines for Successful OGC Interface Standards, OGC document 00-014r1

[2] OGC OWS-5 Conflation Engineering Report, OGC document 07-160r1

[3] OWS 5 SOAP/WSDL Common Engineering Report, OGC document 08-009