Open Geospatial Consortium Inc.

Date: 1.2.2006

Reference number of this OGC® Document: **OGC 05-140r0**

Version: 0.9

Category: OpenGIS® Discussion Paper

Editor: Yves Coene

OWS-3 Imagery Workflow Experiments: Enhanced Service Infrastructure Technology Architecture and Standards in the OWS-3 Testbed

Warning

This document is not an OGC Standard. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an OGC Standard.

Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Document type: OpenGIS® Discussion Paper
Document subtype: Interoperability Experiment Report
Document stage: Approved
Document language: English

Contents

i. Preface

This document describes the results of an experiment addressing issues relating to the application and implementation of workflow experiments utilizing a variety of OpenGIS Implementation Specifications.

This section describes the status of this document at the time of its publication. Other documents may supersede this document. The latest status of this document series is maintained on the OGC website.

This is an OGC Interoperability Program Report (IPR) for review by OGC members and other interested parties. It is a working draft document and may be updated, replaced by other documents at any time. It is inappropriate to use OGC Draft IPRs (DIPRs) as reference material or to cite them as other than "work in progress." This is work in progress and does not imply endorsement by the OGC membership.

This document was developed by OWS-3 participants as part of the OGC Interoperability Program OWS-3 initiative. The authors of this document are OWS-3 participants.

ii. Submitting organizations

This Interoperability Program Report is being submitted to the OGC Interoperability Program by the following organizations:

European Space Agency

Spacebel

Spot Image

University of Alabama Huntsville

iii. Document contributor contact points

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

CONTACT	COMPANY
Yves Coene	Spacebel
The Hoa Nguyen	Spacebel
Philippe Merigot	Spot Image

Alex Robin	UAH

iv. Revision history

Date	Release	Author	Paragraph modified	Description
1.2.2006	0.9	Yves Coene		
3/25/06	0.9	Carl Reed	Numerous	Correct copyright, clarification, etc.

v. Future Work

Future work on the capabilities and issues reported in this document are expected both within the contributing organizations and in subsequent OGC Interoperability Program initiatives.

Foreword

This document describes the results of an experiment addressing issues relating to the application workflow processing incorporating a variety of OGC specifications. It details the inputs provided to the Open Geospatial Consortium's (OGC) OWS-3 Testbed and the architecture of the testbed related to the ESA Service Support Environment (SSE).

It is a formal deliverable of work package 6610 of the Enhanced Service Infrastructure Technology (ESIT) project and is a joint Spacebel and Spot Image document.

This report is informative, and is not a normative (draft or final) OGC Implementation Specification. This report thus does not include any compliance clauses or other specification-specific information.

This report often uses the terminology defined in the OGC Abstract Specification.

Introduction

In March 2005, Spot Image from Toulouse (France) and ESRIN, supported by SPACEBEL, submitted a common "Proposal for participation in the OGC Web Services Initiative Requirement Set 3 and Demonstration". The OWS-3 Testbed activities were kicked-off in April 2005 in Reston (VA) and were completed with a demonstration in Tysons Corner (VA) in November 2005.

The activities performed by SPACEBEL in this framework are part of Phase 1 of the ESIT GSTP Project.

In the ESIT project, technologies are investigated which may either be today applicable in the operational Service Support (SSE) environment or which serve, at this stage, as pure demonstration of future technology.

The current document summarises the result of the activities performed in the framework of the OGC OWS-3 Testbed as a result of this common Spot Image and ESRIN Proposal.

1 OGC Work Context

1.1 WCTS

A Web Coordinate Transformation Service (WCTS) [RD1] exposes an OGC Web service interface to a service that performs coordinate transformations. Such transformations include all the types of coordinate operations, including both "transformations" and "conversions". This service inputs digital features or coverages in one CRS (Coordinate Reference System) and outputs the same features in a different CRS. The service inputs include identifications of the input and output CRSs, and optionally the coordinate transformation between these CRSs.

The WCTS may expose two different interfaces: Key-Value Pair (KVP) and XML encoding. WCTS from different "service providers" are thus not interchangeable. For instance the integration of the Spot Image and PCI service required two different BPEL workflow implementations.

WCTS	*Formats*	*Interface*
Spot Image	GEOTIFF	XML
PCI	GEOTIFF	KVP
GMU	HDF	XML

The status of WCTS at OGC is a "draft Implementation Specification" which is not (yet) an OGC specification.

The WCTS interface as defined by OGC assumes the service is synchronous, which is most often not the case as raster image conversion may be time consuming operation.

1.2 WfCS

According to the Interoperability Program Report from the OWS-2 Testbed [OWS-2], “the Workflow Chaining Service (WfCS) executes workflow processes and correlates and coordinates synchronous interactions into collaborative and transactional business flows. It is an infrastructure service for modelling, connecting, deploying and managing and executing business processes”.

“For each process, the WfCS takes a BPM script (i.e., BPEL “flow”) that describes the workflow or processing chain to be executed, a WSDL document (without binding information) that describes the interface that the process will present to clients (partners in BPEL terms), and WSDL documents that describe the service instances that the process may invoke during its execution. From this information, the process is made available as a Web Service. A WSDL file that describes the process's interface may be retrieved from the WfCS at run-time”.

It seems that the external interfaces of an OGC WfCS are not rigorously defined as other OGC services such as WMS, WFS or WCS and that the WfCS is currently only a loosely defined concept. During the OWS-2 testbed, OGC used the Collaxa BPEL Process Manager, which is an old version of the Oracle BPEL Process Manager available on the ESA SSE Portal.

1.3 SensorML

- Provides (among other things) a description of specific sensor properties (e.g., camera models) that is particularly essential when computationally rigorous image processing operations are applied to coverage data.
- An XML schema for providing sensor system descriptions to support sensor discovery, and geo-location and processing of sensor observations
- Designed to support a wide range of sensors

1.4 OGC WNS

A Web Notification Service [WNS] is a broker-like service used as work-around to implement asynchronous communication. It is being defined as part of the OGC Sensor Web Enablement (SWE) activities.

- Provides a means to alert people, software, or other sensor systems of SPS (Sensor Planning Service) results or alerts regarding phenomena of interest
- An asynchronous and stateful service.
- A web interface that allows sending notifications to a client with well structured content.

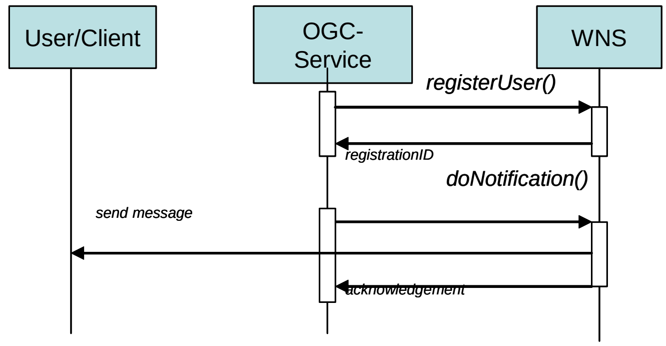

Figure 1: Asynchronous communication via an OGC WNS service

We have not integrated this type of service as it is in our opinion obsolete now the W3C is working on the ws-addressing specification which has a similar objective and is already supported by multiple COTS tools.

In case the "client" is a workflow engine, then software needs to be implemented on the "client" side to be able to process HTTP requests from the WNS and forward the message to the correct workflow instance. This kind of logic is built-in the Oracle BPEL Process Manager for the ws-addressing specification.

1.5 Other Work

1.5.1 WSIF (Apache)

The Web Services Invocation Framework (WSIF) is a simple Java API for invoking Web services, no matter how or where the services are provided. It is based on the Java Community Process (JCP) Java Specification Request (JSR) JSR-110 "Java APIs for WSDL" available at http://www.jcp.org/en/jsr/detail?id=110.

WSIF enables developers to interact with abstract representations of Web services through their WSDL descriptions instead of working directly with the Simple Object Access Protocol (SOAP) APIs, which is the usual programming model. With WSIF, developers can work with the same programming model regardless of how the Web service is implemented and accessed.

The WSIF is part of the WSDL binding framework integrated in the Oracle BPEL Process Manager (See [BPEL]). Oracle BPEL. Process Manager currently supports providers for:

- HTTP GET and POST resources
- Java classes
- EJBs
- JCA

In the future support for JMS will be added.

2 GEOSS Architecture and Demonstration

A first intermediate result was demonstrated at the GEOSS workshop in Seoul (South-Korea) July 24 2005 by G. Percivall (OGC) (see [RD3]) where the SSE was used as a client to start workflows connecting to WCTS and WCS services as depicted below.

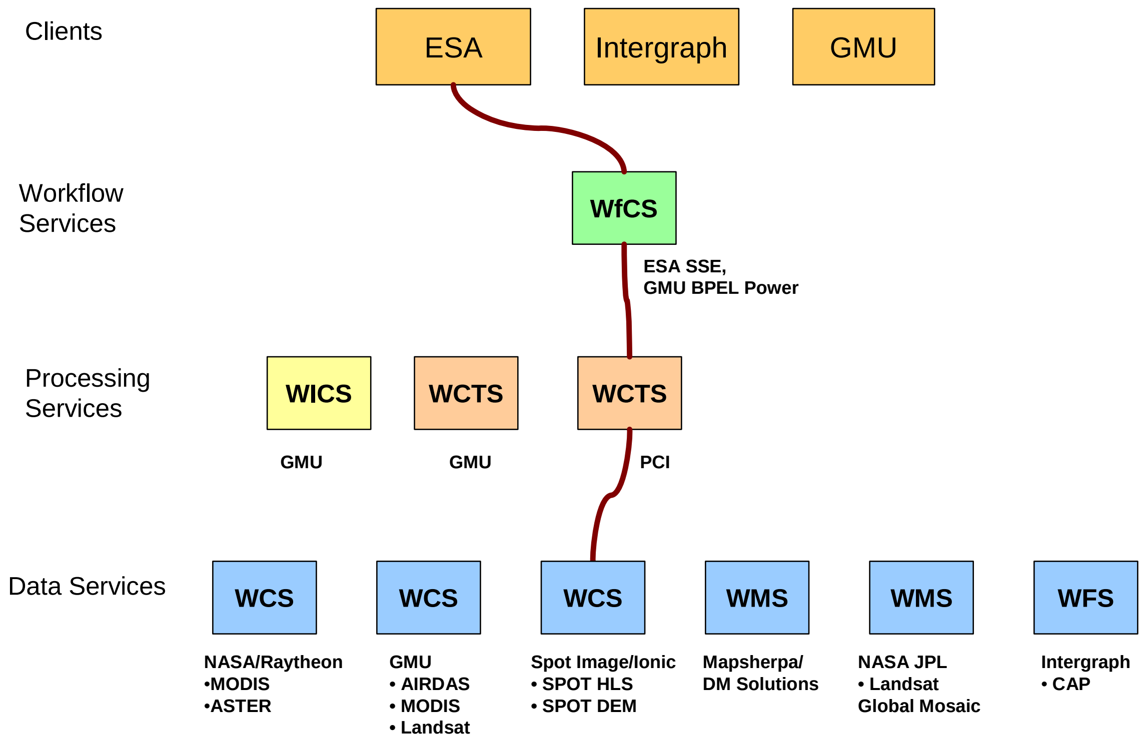

Figure 2: Use of SSE in GEOSS Demo by OGC

For the demo, two services were made available on the SSE Portal under the "ogc" domain. Both are presented in detail below.

Figure 3: OGC's services on the SSE Portal

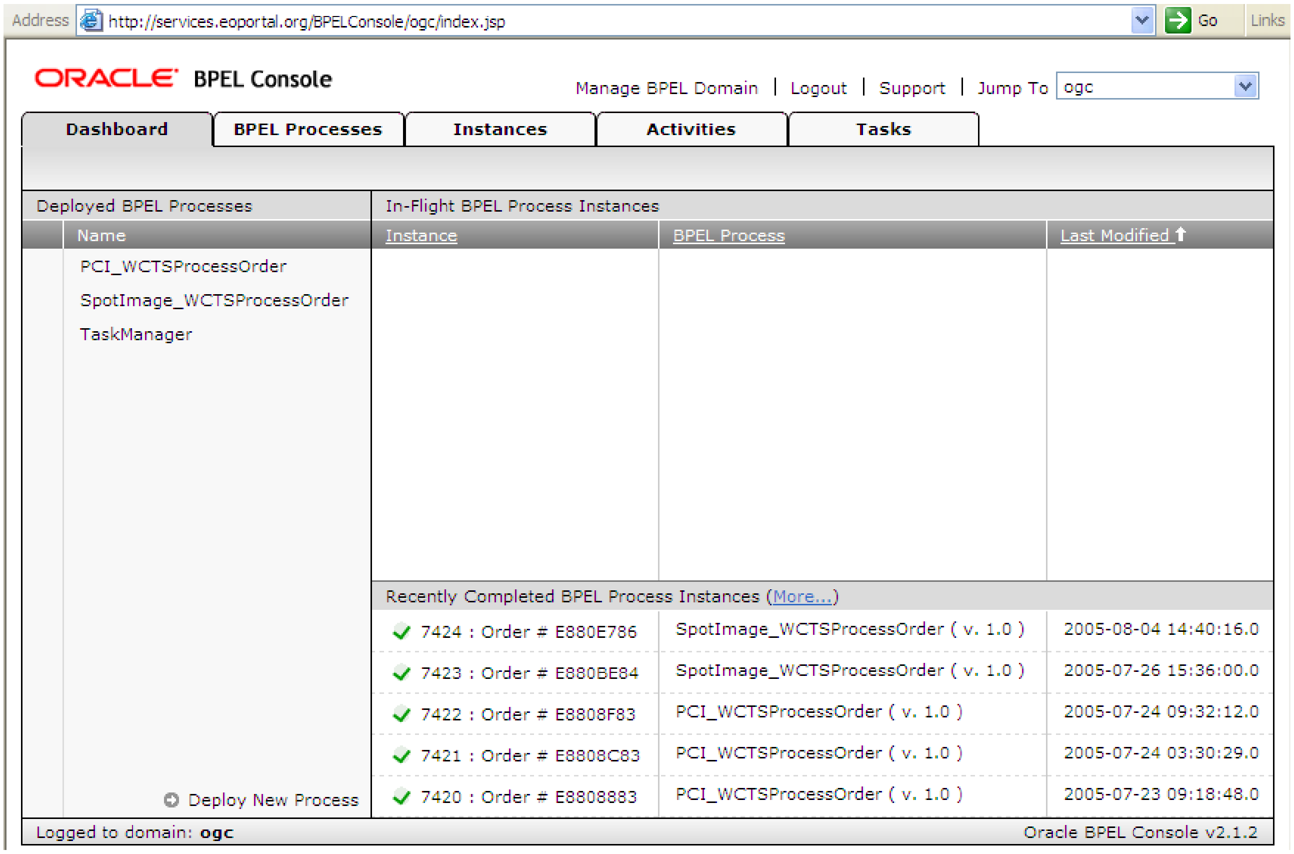

Figure 4: BPEL Processes deployed in OGC domain on SSE Portal

2.1 Spot Image WCS – Spot Image WCTS

2.1.1 WCS

Spot Image has published two WCS services:

- WCS-1 (http://ws.spotimage.com/sisa_wcs_sd) with an image over San Diego
- WCS-2 (http://ws.spotimage.com/wcs) with two images over the Tsunami area, in particular the Dataman Islands. The Tsunami image IDs were given the same identifiers as the identifiers of the images in the Dali catalogue to allow an easier integration later on.

Before Tsunami SSE Product Identifier: 52473270402250437271J2598631, Spot Image WCS identifier: 52473270402250437271J.

After Tsunami SSE Product Identifier: 52473270412280434061J3032572, Spot Image WCS identifier: 52473270412280434061J.

2.1.2 Spot Image WCTS

The WCTS developed for the demonstration is based on the "Tarifa" service [RD6]. "Tarifa" (French for Automatic Image rectification and Fusion Processing) is a service developed by CNES (The French Space Agency) in order to orthorectify any image from SPOT satellites (soon Formosat will be available as well) in an area where Reference3D data is available. The commercial side of this service is called Andorre where Spot Image manages all the customer aspects (feasability, ordering, quality checking, delivery, ...) with a complete integration into their commercial facilities.

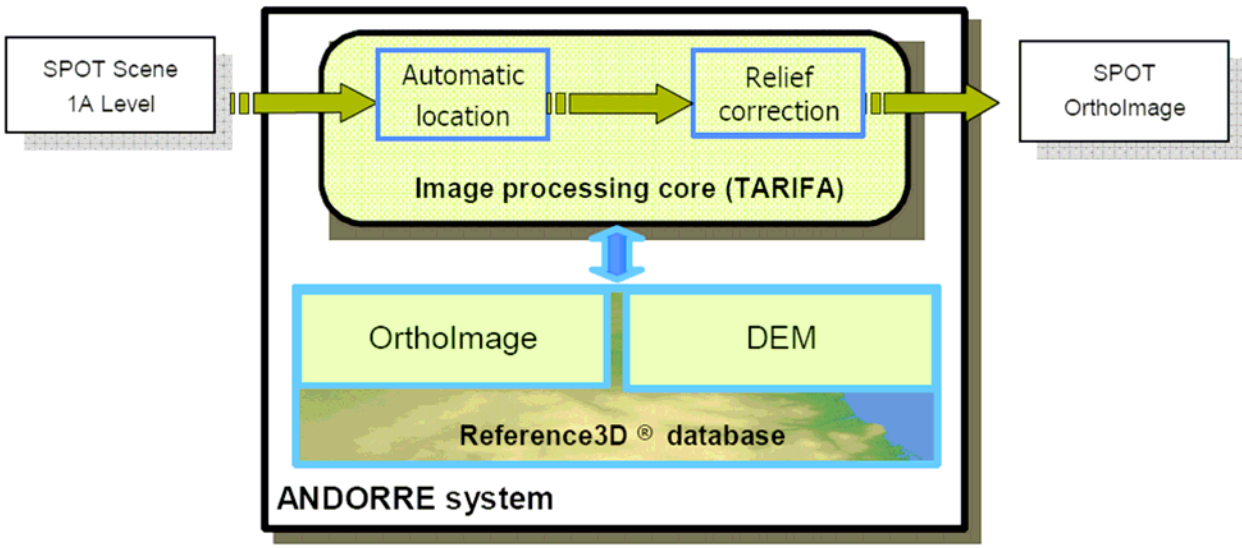

Figure 5: ANDORRE system

For the first demonstration in November 2005 the WCTS will use the DIMAP format to describe all the ancillary information needed by Tarifa and not provided by ISO 19115 part 1. During the preparatory demonstration activity Spot Image will assess the capabilities of SensorML as candidate standard to be use instead of DIMAP.

Spot Image will eventually use a dedicated server with "Tarifa" because it will be separate from their production facilities and easier to maintain. The service will be permanent for a time period to be defined (at least one year) starting from the demonstration day.

- The Spot Image WCTS has an XML over HTTP interface.
- The Spot Image WCTS used for the GEOSS demo was a simulation, always returning the same image.

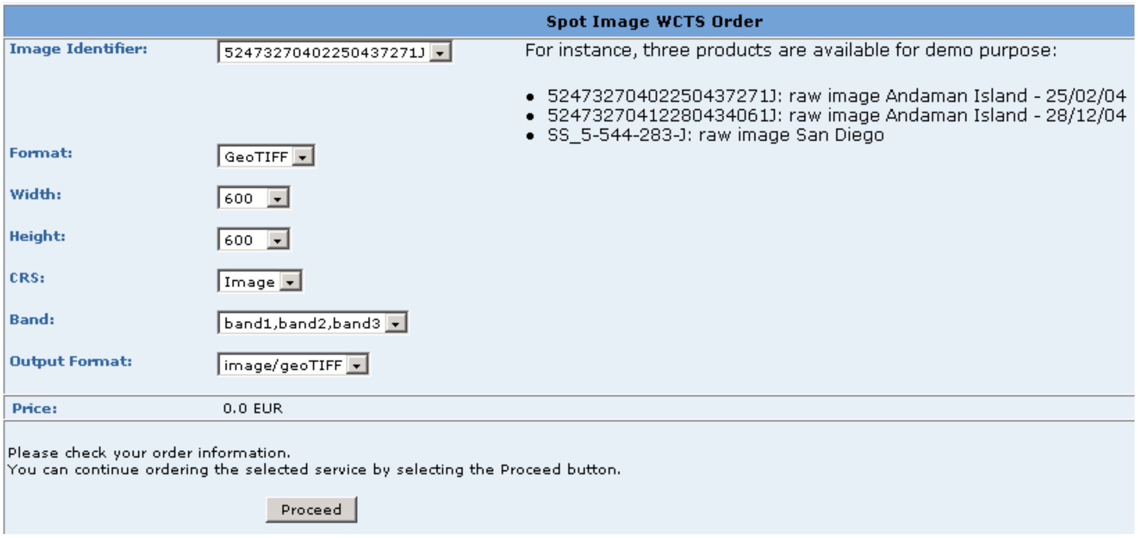

Figure 6: Spot Image WCTS service user interface on SSE Portal

The Spot Image WCTS stores the service result on the Spot Image WCS and returns the URL to retrieve it from the WCS.

The following images are available :

- SPOT5 – nov. 5 2003 - 10m – 4 bands
- SPOT 5 – nov. 22 2003 – 10m – 4 bands
- SPOT 5 – nov. 22 2003 – 10m – panchromatic
- SPOT4 – nov. 18 2003 – 20m – 4 bands

2.1.3 BPEL Workflow

The BPEL workflow takes as input the data from the order page depicted in Figure 6.

To avoid receiving the coverage data returned by the WCS in the workflow, we do not call the WCS, but assemble a correct URL for the coverage which is passed as a parameter to the WCTS service.

As the WCTS service has no SOAP bindings, but uses XML over HTTP Post, we reverse engineered a WSDL file for the WCTS by investigating the HTTP messages exchanged with the service. The resulting WSDL file is included in section **8.1.1.2**. It uses the WSIF for accessing services with non-SOAP bindings.

The WCTS needs a DIMAP file to be able to process the GEOTIFF input image. The URL of the image is returned by the SIAS (Spot Image Archive Service). The DIMAP files reside on a Spot Image server.

Further work: In operational workflows, it would be better if the DIMAP URL would be part of the image metadata and possibly included in an order result message.

The resulting workflow executes correctly on the Oracle BPEL Process Manager version 2.1.2. It can be validated with the BPEL Designer.

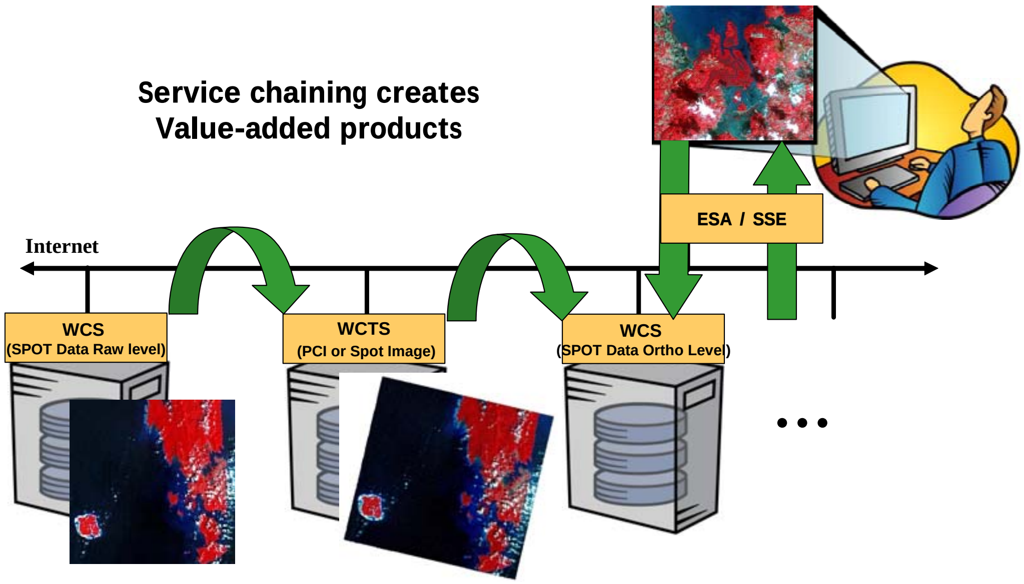

Figure 7: SSE as client to start the workflow (source: GEOSS presentation OGC)

2.2 Spot Image WCS – PCI WCTS

2.2.1 WCS

See section 2.1.1.

2.2.2 PCI WCTS

The PCI (www.pcigeomatics.com) WCTS uses an HTTP Key-Value Pair (KVP) interface. A test page was available at http://gws2.pcigeomatics.com/wcts/index.html at the time of the GEOSS demo. The contact person was Steve Keens (keens@pcigeomatics.com).

The service had a number of limitations at the time of the GEOSS demo. (personal communication 14/07/05 from Steven Keens). For instance, it cannot handle concurrent requests. To avoid problems during the demo, the service was put in test mode on the SSE Portal.

2.2.3 BPEL Workflow

The BPEL workflow takes as input the data from the order page depicted in Figure 8.

To avoid receiving the coverage data returned by the WCS in the workflow, we do not call the WCS, but assemble a correct URL for the coverage which is passed as a parameter to the WCTS service.

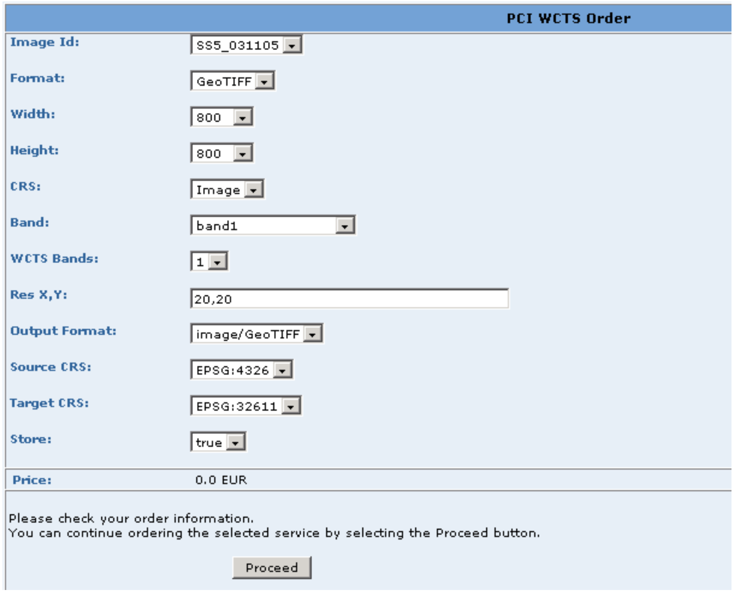

Figure 8: PCI WCTS service user interface on SSE Portal

As the WCTS service has no SOAP bindings, but uses a KVP interface, we reverse engineered a WSDL file for the PCI WCTS by investigating the HTTP messages exchanged with the service. The resulting WSDL file is included in section **8.1.1.2**. The WSDL file shows that the service bindings are not SOAP bindings, but instead use the WSIF which is part of the BPEL Process Manager.

3 Limitations of GEOSS Demo Architecture

1.1 Connection with Dali Catalogue

The two BPEL workflows implementing an Order operation for a WCTS on the SSE which were presented above in sections 2.1 and 2.2 need as input a product identifier, which is selected from a drop-down list as is shown on Figure 6 and Figure 8. In this drop down-list (which is generated by the service XSLT stylesheet), the product IDs are listed which correspond to images actually available on the WCS. There is currently no link between the Spot archive and the WCS.

In an operational service, the product selection should not be done from a drop-down list, but via a DALI catalogue search. Figure 9 shows that the products used during the GEOSS demo can indeed be found via the DALI catalogue interface available on the SSE Portal (service "Products", organisation "ESA").

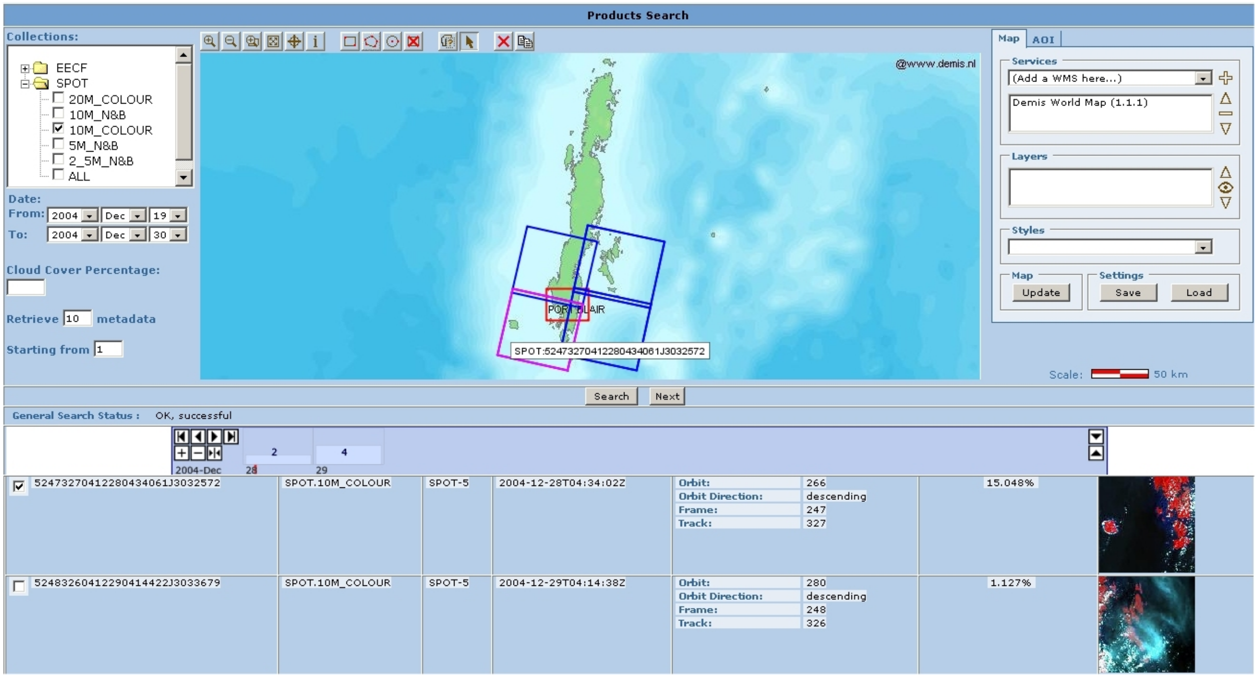

Figure 9: User interface of Dali catalogue on SSE Portal showing Tsunami (after event) imagery

3.1 Connection with Order

The Spot Image WCS only gives access to a very limited test set of images. There is no automated link with the Spot Image archive. To allow users to select an arbitrary image from the catalogue and use that as input for the WCTS, a manual operator intervention is need to allow an operator at Spot Image to physically copy an image from the archive to the WCS server. This interaction is logically part of the product "Order" operation for Spot Image images.

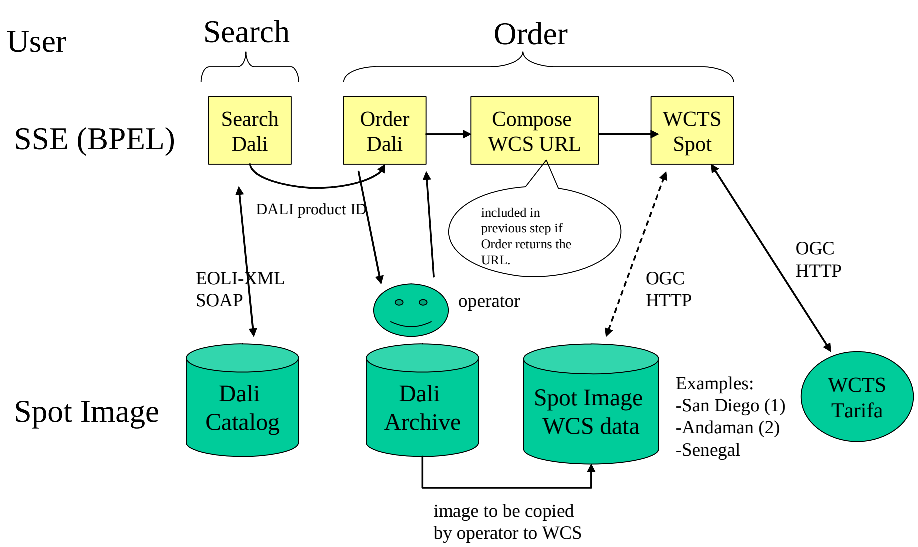

Figure 10: Search followed by Order on SSE Portal

In the figure above, the yellow boxes represent parts of the BPEL workflow running on the SSE's Oracle BPEL Process Manager in Frascati (Italy). Green elements are located at Spot Image in Toulouse (France).

3.1.1 Possible Implementations

The Order operation should send at least the identification of the product and return the WCS URL of the product, after the operator having transferred the image to the WCS storage. This asynchronous communication with the Spot Image operator can be implemented in a number of ways which have been discussed with Spot Image. The options are briefly described below.

3.1.1.1 Option 1: Two-way Email

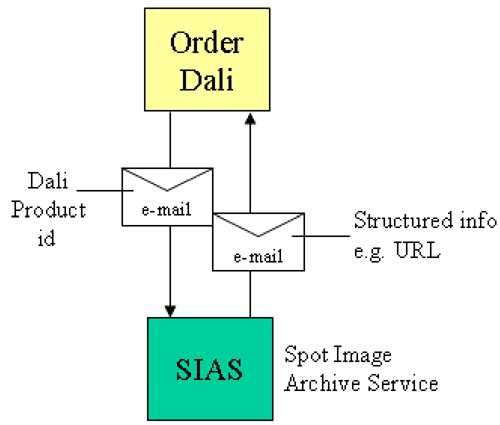

Figure 11: Two-way Email

A first option is to communicate between the BPEL workflow and the Spot Image ordering interface using emails. A BPEL workflow instance can send a structured email and wait until it receives an email. The advantage of this implementation is that it requires no implementation on the Spot Image side. An operator can receive and create the emails using his email client (Microsoft Outlook, Mozilla Thunderbird etc.)

3.1.1.2 Option 2: One-Way Email and User Task

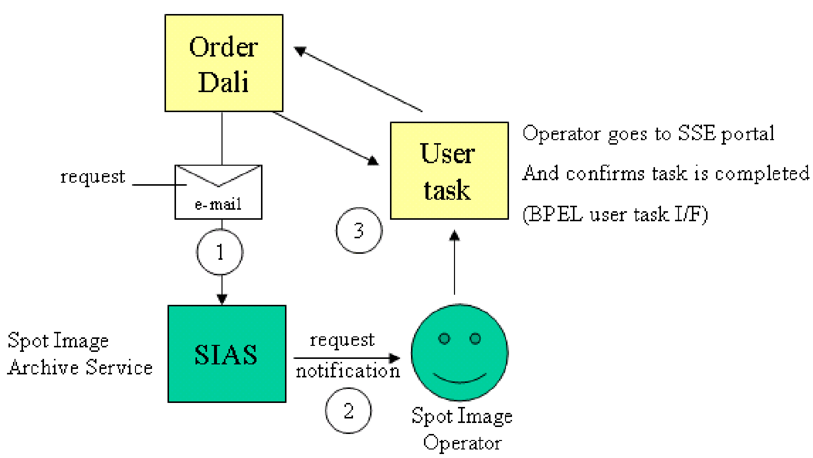

Figure 12: One-way Email and User Task

A variation on the first alternative is to replace the email which is sent back to the BPEL instance with a user task. The BPEL workflow sends an email to Spot Image containing the order details. The operator prepares then the order and uses the User Task user interface in the Oracle BPEL Console to indicate the order is ready. The BPEL instance resumes when the user task is completed, and then calls the next step in the chain which is the WCTS service.

Option 3: OGC Notification Service

OGC has defined a work-around to allow for asynchronous communication using only synchronous messages.

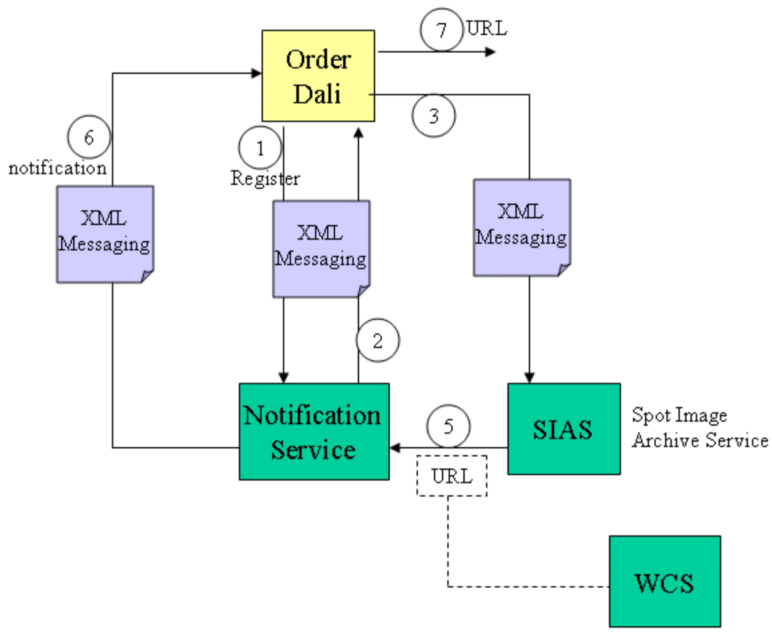

Figure 13: OGC Notification Service for asynchronous communication

Option 3 is an implementation using the OGC Notification Service concept. It comprises the following steps:

1. The client (Order Dali) registers to the NS (Notification Service). The registering message contains the way(s) the client wants to be notified (e-mail, SMS, POST message, etc).
2. The NS sends back a unique NID (Notification ID) to the client.
3. The client queries the SIAS (Spot Image Archive Service). The NID is transmitted.
4. The SIAS processes the Query.
5. The SIAS sends a message containing the result (WCS URL) and the NID to the NS.
6. The NS identifies the client with the NID and sends a notification containing the result to the address(es) given by the client when he registered.

There are a number of disadvantages with this approach:

1. The integration with the BPEL workflow which is the client of this asynchronous service is not trivial. BPEL contains the necessary support for a workflow to wait until an asynchronous operation terminates using ws-addressing. For this ad-hoc OGC mechanism, there is no support in BPEL.

2. The interfaces do not use SOAP which is the preferred protocol to be used with BPEL.

3.1.1.3 Option 4: Asynchronous SOAP with ws-addressing

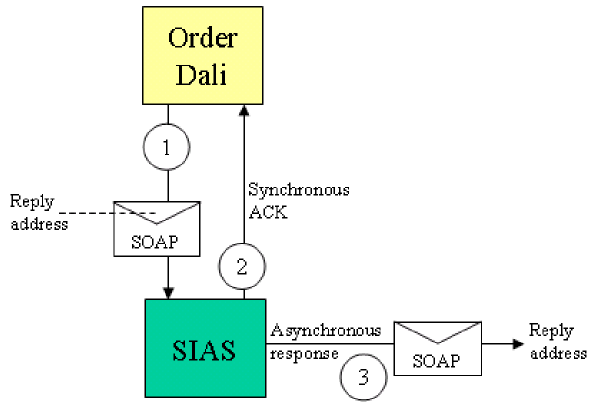

Figure 14: Asynchronous SOAP communication

This implementation option consists of the following steps:

1. The client BPEL workflow (Order Dali) sends a SOAP message to the SIAS (Spot Image Archive Service). The header of the SOAP message contains the reply address, the body contains a query.
2. The SIAS sends back a response synchronously to tell the client whether it will process the query or not.
3. In case the SIAS has processed the query a SOAP message containing the result is sent to the reply address found in the header of the SOAP message (using ws-addressing) sent by the client.

This option has been selected for implementation as it uses a standard method for asynchronous communication, i.e. ws-addressing from W3C. This allows the service to be called from BPEL. The details of the implementation are given in chapter **4**.

Spot Image preferred to use SOAP with attachments instead of normal SOAP for the following reasons :

- The Spot Image WCTS provides two methods for querying a transformation :
  1. XML messaging via http/Post
  2. SOAP
  By using SOAP with attachment it is easier to test a client which implements both methods because the requests are stored in the same files. It's also easier to test the service because the code is the same for both methods.
- The client does not need to know which method is used. He focuses only on writing an XML file containing the request according to the schema.
  Moreover if his request is not valid, the XML file containing the request is the only thing he has to modify. It can also create several files containing different requests.

- It is normal practice to include XML data in the SOAP envelope but using attachment is not disadvised. The transform request can be complex and big. Sending it as an attachment separates the transport and the content which can be compressed and/or encrypted for instance.

We decided to use normal SOAP because of the following reasons:

- Oracle has confirmed that the BPEL Process Manager 2.1.2 and 10.1.2 do not support SOAP with attachements.

It is normal practice to include XML data in the SOAP envelope instead of encoding it as an attachment.

4 OWS-3 DEMO ARCHITECTURE SPOT IMAGE – ESA

4.1 Overview

This is the design for the joint ESA and Spot Image OWS-3 testbed activity. The main improvements w.r.t. the GEOS demo architecture are:

- The WCTS service is now a real implementation based on the CNES TARIFA software.
- The WCTS is now an asynchronous service instead of a synchronous service.
- The WCTS is integrated with an ordering function (called SIAS by Spot Image) and a Dali catalogue search.

The sequence of a typical user interaction is shown below in UML.

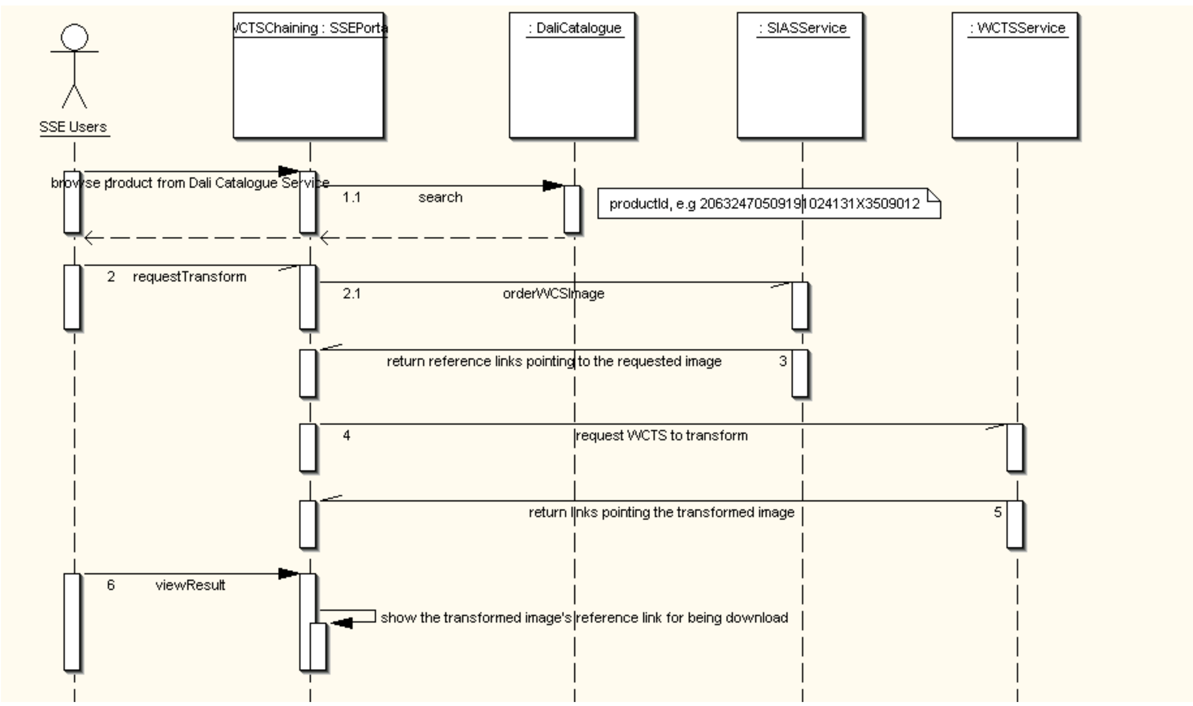

Figure 15: Sequence diagram interaction with WCTS workflow on SSE Portal

4.2 SSE User Interfaces

The Web pages giving access to the workflow from the SSE Portal (http://services.eoportal.org) are shown below. The service is registered as a Spot Image service and is called “WCTS Chaining”.

Figure 16: Service Description on SSE Portal

As the service is asynchronous, it may take a while before the result becomes available. You can define in your SSE profile (see screen dump below) that you would like to receive an email when the result of asynchronous services becomes available in your SSE order list.

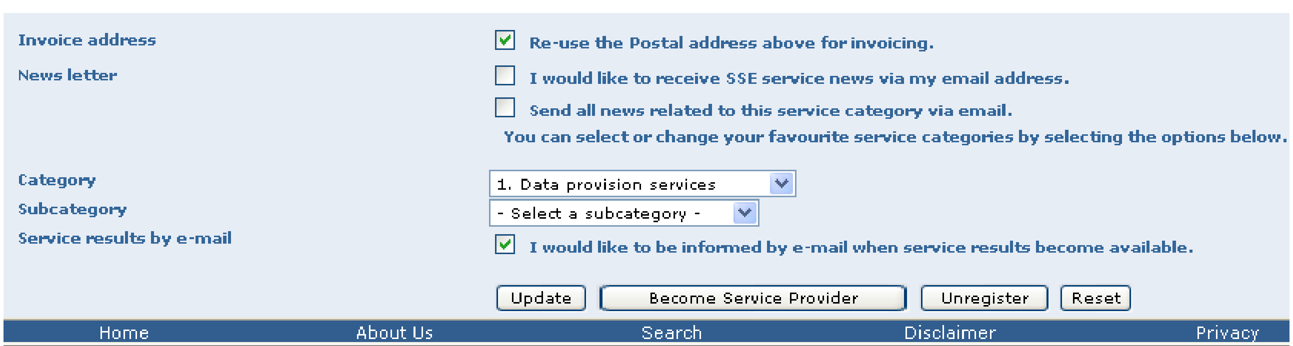

Figure 17: SSE User Profile Page

4.2.1 Catalogue Search

The first step of this service is to select an image to be processed by the WCTS service. This is done by selecting the image in the Spot Image Dali catalogue. As only very limited data are available in the WCS for the demo, the possible dates and the area of interest the user is allowed to select are restricted. The area is limited to San Diego.

The Dali catalogue SOAP interface is implemented according to OGC Discussion Paper 05-057, previously called the ESRIN "EOLI-XML" ICD.

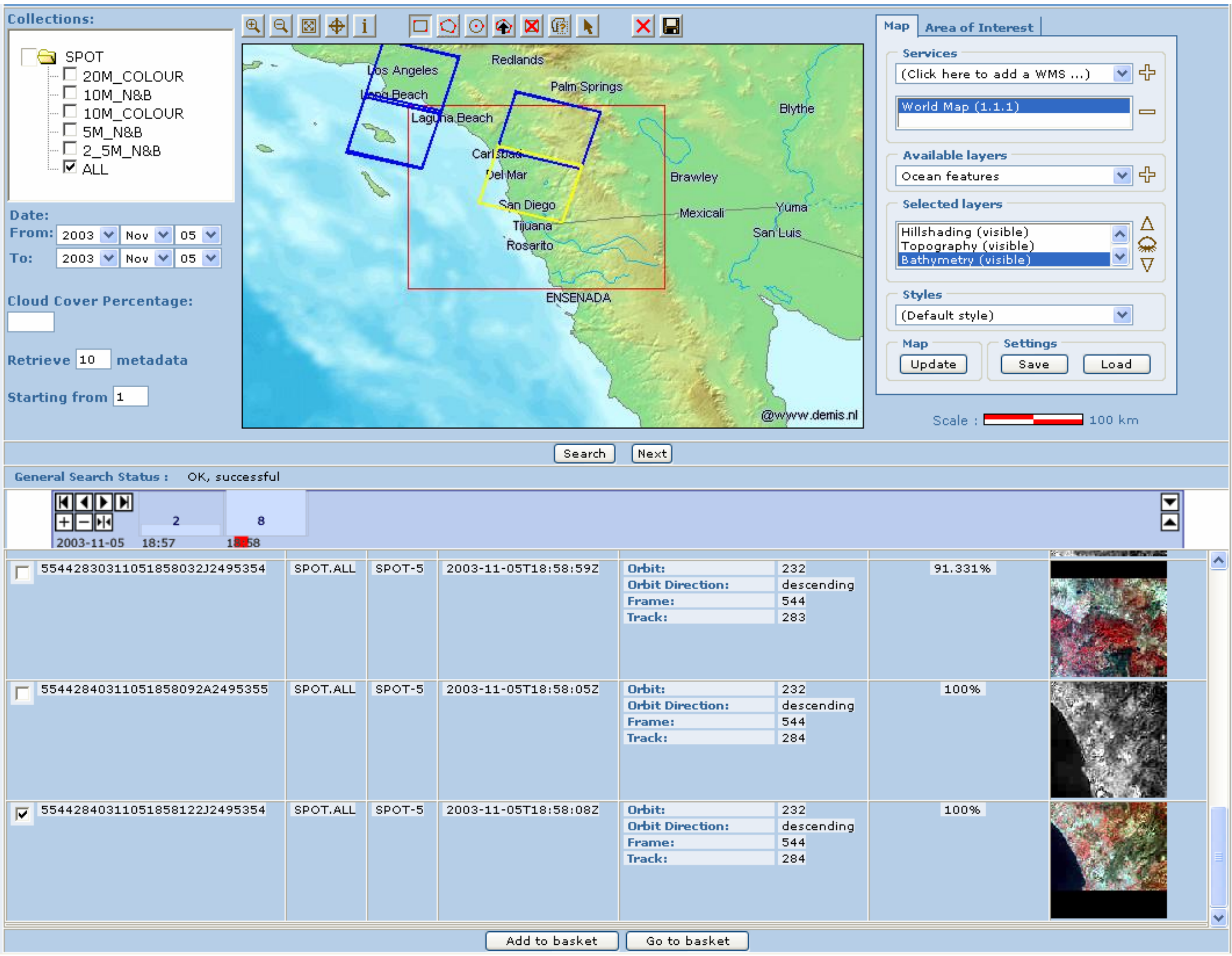

Figure 18: Dali Catalogue Search

4.2.2 Order

The second step is the activation of the ordering. This is done by selecting the search result on the catalogue search result page. In the picture below, the image over San Diego was selected. On the left side of the page, the user can select the WCTS parameters he wants to use for the service.

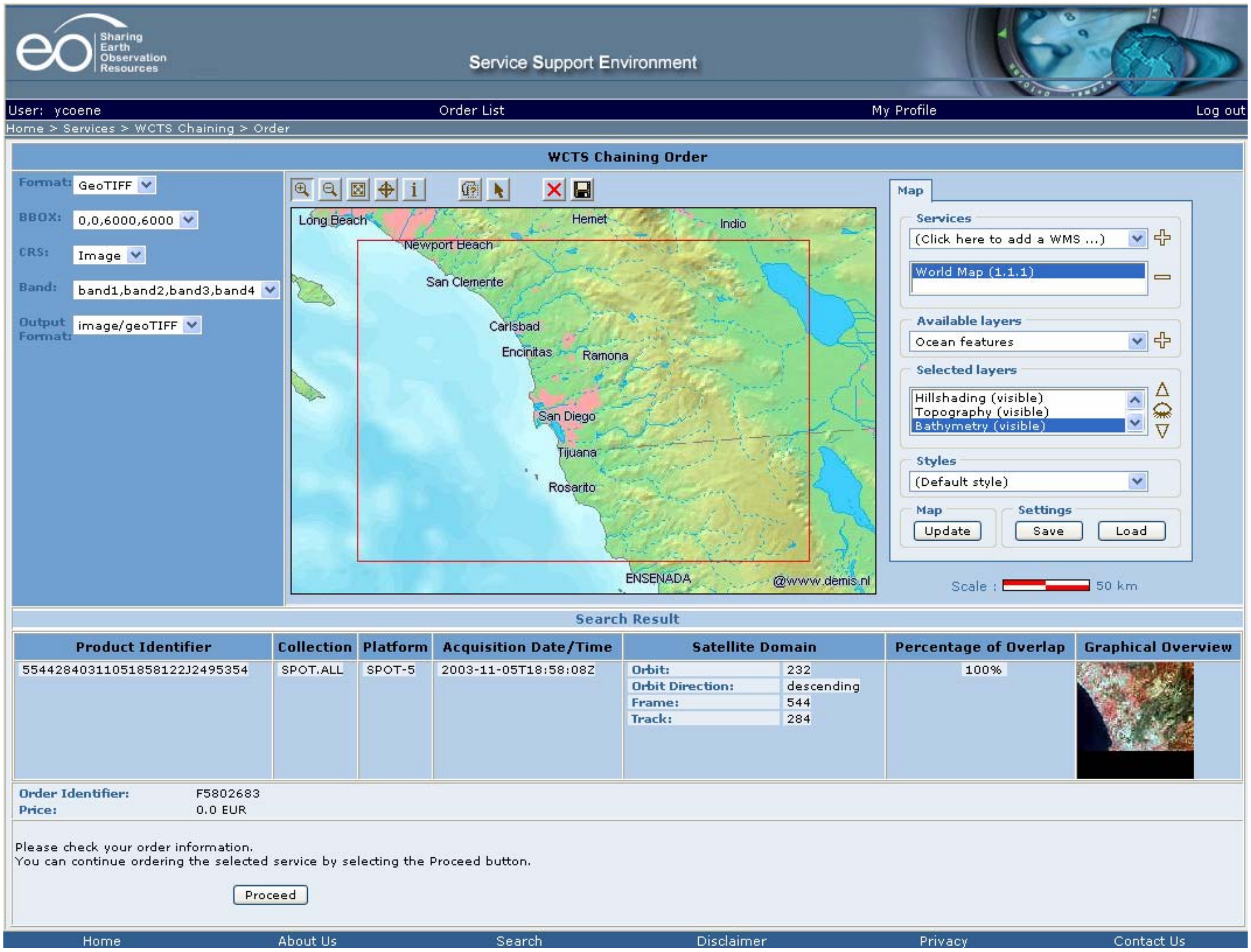

Figure 19: Ordering Interface

4.2.3 Access to Service Result

Once the service is ordered, its result will become available via the user's order list. The status of the service in the order list changes from "pending" to "completed" when the result is available. The user also receives an email when the service result is available if he had set the corresponding option in his SSE user profile.

Figure 20: SSE Order List

Selecting the Order ID corresponding to the order gives access to the service result page. On the page, the transformed image and the original image are both accessible via a hyperlink as shown below.

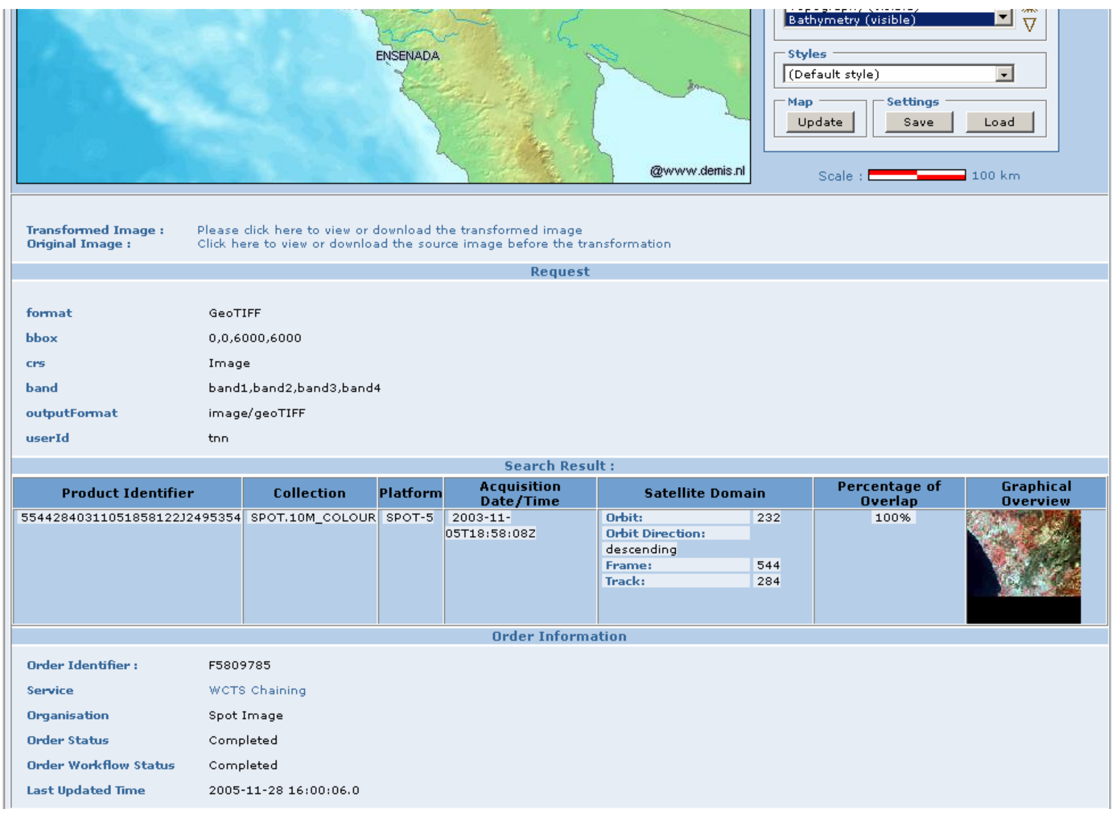

Figure 21: SSE Order Result Page

4.3 Service Workflow Design

The BPEL source code of the workflows is available in section **8.2**. One of the requirements was to allow access to the basic WCTS service from two different clients:

- SSE Portal.
- OWS-3 Decision Support client(s).

The two workflows allowing access to the WCTS are different as the two clients use different interfaces with the workflow engine:

- SSE Portal starts a workflow via an API and receives results via a JMS queue.
- Other clients can invoke the workflow via SOAP and receive the results via SOAP and ws-addressing.

The workflow is therefore developed in separate BPEL modules:

- WctsFlow0.bpel: This workflow is the only one which accesses directly the Spot Image WCTS service. Changes to the WCTS interface have thus only an impact on this BPEL file. It is not accessed by any client.
- WctsFlow1.bpel is called by the SSE Portal, it calls WctsFlow0
- WctsFlow2.bpel is called by SOAP clients, it calls WctsFlow0. Its interface is published as wctsFlow.wsdl and documented in section **8.2.1.5**.
- SIASFlow0.bpel: This workflow is the only one which accesses directly the Spot Image ordering interface. It also returns a WCS URL allow the client to access the raster image via the Spot Image WCS server.
- ChainingFlow.bpel is the BPEL process implementing the flow combining the ordering service “sias” and the WCTS service. It calls internally the WctsFlow0.bpel and SIASFlow0.bpel modules. It is called by the SSE Portal.

4.3.1 Interface for SOAP client

WctsFlow2.bpel can be called by SOAP clients, it calls WctsFlow0.bpel. Its interface is published as wctsFlow.wsdl and documented in section **8.2.1.5**.

4.3.2 Interface for OGC Web service client

OGC type of Web services do not use SOAP bindings but HTTP Get or Post. A similar interface can also be published via the Oracle BPEL Process Manager. An implementation of this will be made and documented in a future version of this Technical Note.

According to Oracle it would be sufficient to copy the binding part of a WSDL file with OGC type bindings in the empty binding part of the WSDL file corresponding to the published workflow. This will be tested and documented in detail in a future version of this Technical Note. An example of a flow exposing such kind of binding is available as example 7-02 (TBC) in the Oracle BPEL 10.1.2 distribution package.

4.4 WCTS Implementation at Spot Image

4.4.1 Design

The design comes form the following constraints:

- the orthorectification process is not accessible from the Internet and is not installed on a computer which have an access to the Internet;
- the orthorectification process and the WCTS may run on different operating systems;
- the orthorectifiaction process can be replaced by another one, eventually implemented by a different company with a different technology, without any modification into the WCTS.
- The WCTS must be able to handle several requests at the same time but the current orthorectification process can handle only one at the same time.

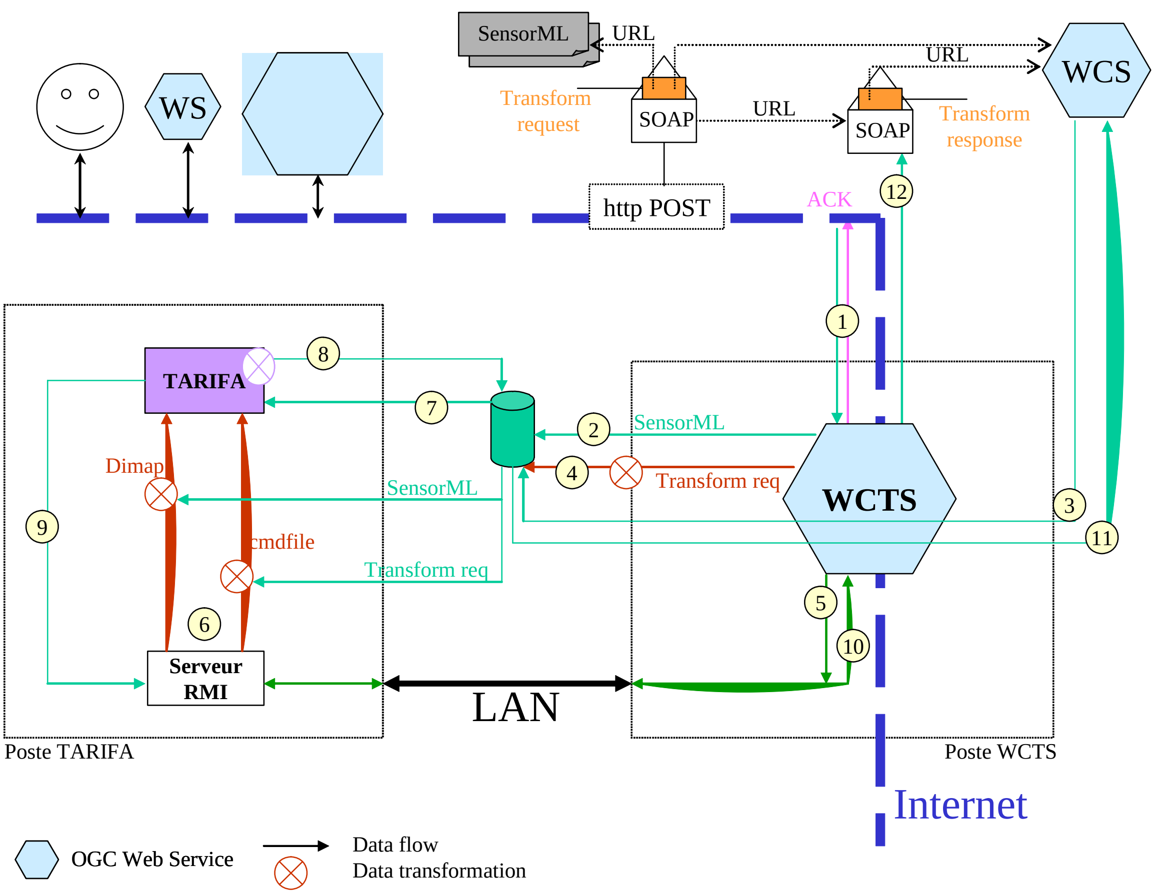

Figure 22: Architecture for WCTS at Spot Image (source: P. Merigot)

1. A client sends a SOAP message to the WCTS. The header of the SOAP message contains a reply address for asynchronous response and the body contains a Transform request encoded in XML. The WCTS checks the parameters and sends a synchronous response to indicate it will process the request or not. The requests are processed in separate threads.
2. The metadata files (Dimap or SensorML) are downloaded and stored locally.
3. The image to be transformed is downloaded and stored locally.
4. The Transform request is extracted from the SOAP message and stored locally. The fields containing the URL to the metadata files and the image are modified to point to the local copy.
5. The WCTS connects to the RMI Server.
6. The RMI server converts the Transform request into a proprietary command file. The SensorML files are converted into Dimap by using a stylesheet. See section **8.3** for a DIMAP example file and the corresponding SensorML files.
7. The image is processed.
8. The resulting image is copied locally, somewhere accessible by the WCTS.
9. The RMI server is notified when the orthorectification is done.
10. The RMI server notifies the WCTS that the process is done and indicates where the resulting data is stored.
11. The WCTS puts the resulting image into the WCS.
12. The WCTS sends a SOAP message to the reply address (cf. step 1). The body of the message contains the URL to the WCS where the resulting image is available.

4.4.2 Request Example

The following code shows a SOAP message containing a Transform request:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope                          xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"       xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
 <soapenv:Header>
   <ns1:MessageID            ns2:rootId="9213"            ns2:parentId="9213"            ns2:priority="3"
xmlns:ns1="http://schemas.xmlsoap.org/ws/2003/03/addressing"
xmlns:ns2="http://schemas.oracle.com/bpel">bpel://localhost/spotimage/wctsFlow0~1.0/9213-BpInv0-
BpSeq0.3-3</ns1:MessageID>
   <ns3:ReplyTo xmlns:ns3="http://schemas.xmlsoap.org/ws/2003/03/addressing">
     <ns3:Address>http://ws.spotimage.com/SOAPReceiver/SOAPReceiver</ns3:Address>
     <ns3:PortType xmlns:ns4="http://www.opengis.net/wcts">ns4:WCTSCallback</ns3:PortType>
     <ns3:ServiceName
xmlns:ns5="http://www.spotimage.fr/wcts">ns5:WCTSCallback</ns3:ServiceName>
   </ns3:ReplyTo>
 </soapenv:Header>
 <soapenv:Body>
   <sendOrderInputMsg    xmlns="http://www.opengis.net/wcts"    xmlns:mass="http://www.esa.int/mass"
xmlns:ns1="http://www.opengis.net/wcts">
```



```
      <commonInput xmlns="http://www.esa.int/mass">
        <orderId>123</orderId>
      </commonInput>
      <sendOrderInput>
        <Transform xmlns="http://www.opengis.net/wcts" xmlns:gml="http://www.opengis.net/gml"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:ows="http://www.opengis.net/ows" service="WCTS"
version="0.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/wcts wctsTransform.xsd">
          <transformation>
            <Transformation xmlns="http://www.opengis.net/gml" gml:id="orthorectification_spot">
              <coordinateOperationName>orthorectification</coordinateOperationName>
              <operationVersion>1.0</operationVersion>
              <sourceCRS/>
              <targetCRS/>
              <usesMethod/>
              <usesValue>

<valueFile>http://ws.spotimage.com/sisa_wcs_sd/coverage/SS5_031105/REQUEST/getdir/DIR/metadata
/DATA/LPR/SS5_031105/METADATA.DIM</valueFile>
                <valueOfParameter xlink:href="http://www.spotimage.com/wcts#Dimap"/>
              </usesValue>
            </Transformation>
          </transformation>
          <InputData>
            <Reference
xlink:href="http://ws.spotimage.com/sisa_wcs_sd/coverage/SS5_031105?request=GetCoverage&service
=WCS&version=1.0.20&COVERAGE=TIF&store=false&CRS=Image&RESPONSE_CRS=image&BBOX=
0,0,6000,6000&WIDTH=6000&HEIGHT=6000&measurename=band1,band2,band3,band4&FORMAT=Ge
oTIFF&EXCEPTIONS=application/vnd.ogc.se_xml"/>
          </InputData>
          <Output>
            <Format>image/GeoTIFF</Format>
          </Output>
        </Transform>
      </sendOrderInput>
    </sendOrderInputMsg>
  </soapenv:Body>
</soapenv:Envelope>
```

4.4.3 Response example

The following code shows a SOAP message containing a response to a Transform request:


```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header>
    <wsa:RelatesTo xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing" xmlns:xsd="www.w3.org/2001/XMLSchema" xmlns:xsi="www.w3.org/2001/XMLSchema-instance" SOAP-ENV:mustUnderstand="0" xsi:type="xsd:string">bpel://localhost/spotimage/wctsFlow0~1.0/9213-BpInv0-BpSeq0.3-3</wsa:RelatesTo>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <n:returnOrderResultInputMsg xmlns:mass="http://www.esa.int/mass" xmlns:n="http://www.opengis.net/wcts" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <mass:commonInput>
        <mass:orderId>123</mass:orderId>
      </mass:commonInput>
      <n:getOrderOutput>
        <mass:statusInfo>
          <mass:statusId>0</mass:statusId>
          <mass:statusMsg>Successful</mass:statusMsg>
        </mass:statusInfo>
        <TransformedData xmlns="http://www.opengis.net/wcts" xmlns:xlink="http://www.w3.org/1999/xlink">
          <Reference xlink:href="http://ws.spotimage.fr/sisa_wcs_res/coverage/123_5287818801234638002?request=GetCoverage&service=WCS&version=1.0.20&COVERAGE=IMAGERY&store=false&FORMAT=GeoTIFF&CRS=EPSG:32611&RESPONSE_CRS=EPSG:32611&WIDTH=6000&HEIGHT=6000&BBOX=456050.0,3623290.0,545080.0,3701540.0&measurename=band1,band2,band3"/>
        </TransformedData>
      </n:getOrderOutput>
    </n:returnOrderResultInputMsg>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```


5 Conclusions

5.1 Lessons Learnt

- The WCTS schema is sometimes unclear. For instance the WCTS needs a URL for the image to be transformed and one or more URLs for the metadata. The URL for the image is encoded into the *Transform/InputData/Reference* element. Should the URLs to the metadata be encoded into the same element (which can occur to infinity) or into *Transformation/metadataProperty/_Metadata* or into *Transformation/usesValue/valueFile* ?
  In which case should we use *CoordinateOperation*, *SingleOperation*, *Operation*, *GeneralTransformation* or *Transformation* ?

- For better interaction between the user and the service we implemented a few commands, which can be sent to the service by using a KVP http/Get query :

- **status** : allows the user to know the status of an order (waiting, running, completed, cancelled);
- **cancel** : allows the user to cancel an order;
- **help** : display the list of available commands with their parameters;

5.2 Future Work

5.2.1 Replacement of DIMAP by SensorML

A stylesheet for automatic conversion from SensorML to Dimap has been integrated into the WCTS. When the WCTS recognizes the metadata file as a SensorML file the stylesheet is automatically executed to produce a Dimap file.

Unfortunately Spot Image is still waiting for inputs of UAH in order to finalize the stylesheet. Therefore this feature is not operational for the moment.

5.2.2 Service Registry

The workflows could be registered in a service registry based on either OGC CSW 2.0, UDDI or ws-inspection.

It may be envisaged to allow harvesting workflows in an OGC 2.0 testbed catalogue.

This idea was raised by the OGC CA group (teleconf 8 august 05) to describe the workflows in a Capabilities document using “OWS Common”. This could be a separate static document supplied when the service is added to the catalogue. See document 05-008.

5.2.3 Other Improvements

The following table list possible ameliorations to the current implementation.

N°	Priority	Action	Title	Description	Cost SI	Cost spacebel	Comments
1	1 (high)	ESA - SI	Image not available through the WCS	If the image selected by the user is not available through the WCS the portal allows the user to connect to an ordering service* : - step 1 (low cost) : when invoked, the ordering service sends a notification to the client service of Spot Image; - step 2 (high cost) : the ordering service transmits the commands to the Spot Image production service via Gescom/Gesprod; * In a real situation the user might invoke an SPS for a new acquisition	- low - high		
2	1	ESA - SI	Transformation parameters	The user should be able to set all the parameters of the transformation : bounding box, targetCRS, etc. TBD : all the parameters that Tarifa can handle.	high	high	

3	1	ESA	The SSE builds the WCS query	For the moment the WCS query is built by the SIAS. The user may want to choose the Bounding Box through the portal for example, the query will change. Therefore the WCS query should be build by the SSE.	low	high	This would mean a user wants to order a subset of an image and would require an interface where a user can select a subset of an image returned by the catalogue.
4	1	ESA - DI	Optional operation requests	The optional operation requests are : IsTransformable, GetTransformation, DescribeTransformation, DescribeCRS, DescribeMethod (cf. RD1).	high	/	To be analysed whether these operations need to be supported by the SSE client.
5	2 (mid)	SI	Sensor description	The sensor is currently described in a dimap file. In a multi mission context,.should it be described in a generic format (sensorML) ?	high	/	[Spacebel] See section **Error! Reference source not found.**
6	2	ESA - SI	Securing access	On the SSE side, transmit the login/password to the WCTS. On the WCTS side, check authorizations and transmit login/pwd to the WCS.	mid		[Spacebel] Spacebel recoimmends to implement a more general solution not specific for this WCTS service using ws-security or using a non-intrusive Web service security solution. [SI] The login/pwd must be stored in the SOAP message, not in the WCTS transform request.

7	2	ESA - SI	Order status	On the SSE side, the status order is *pending* or *completed*. On the WCTS side, the status order is *running*, *waiting*, *completed* or *canceled*. The information about the status order should be shared between the SSE and the WCTS.	mid or high		[Spacebel] The SSE only shows the status of the workflow. BPEL scope names can be used to show intermediate statuses. If more intermediate statuses are needed, then the ordering protocol has to allow this. To be seen whether the HMA-I Ordering ICD will allow this.
8	2	ESA - SI	Error messages	When a problem occurs, the user should be able to understand the cause, especially if the parameters given by the user are wrong.	high	- low	
9	2	ESA - SI	Choice of rectification process	It is possible to allow the user to choose between N2 (fast, works with all images) and N3 (very slow, special data required). In case of N3, the service must check if the required data are available before processing.	- low - high	- low - mid	[SI] : the *usesMethod* element of the Transform request can be used to tell the service which method (N2 or N3) should be used.

6 BPEL Workflows

Describe how your analyzed the data and present the results of that analysis. This section must have sufficient detail that a third party can validate your analysis using the data provided in paragraph 4.

7 Conclusions

Summarize lessons learned and conclusions drawn from this experiement..

8 Terms and definitions

For the purposes of this document, the following terms and definitions apply:

BPEL	Business Process Execution Language
CNES	Centre National d'Etudes Spatiales (French Space Agency)
DIMAP	Digital Image Map
ESIT	Enhanced Service Infrastructure Technology
GEOSS	Global Earth Observing System of Systems
GMU	George Mason University
KVP	Key Value Pair
OGC	Open Geospatial Consortium
OWS	OGC Web Services
RFW	Request For Waiver
SSE	Service Support Environment
SWE	Sensor Web Enablement
TBC	To Be Confirmed
TBD	To Be Defined
WCS	Web Coverage Service
WCTS	Web Coordinate Transformation Service
WNS	Web Notification Service
WSDL	Web Services Description Language
WSIF	Web Services Invocation Framework
XML	eXtensible Markup Language

Bibliography

Statement of Work GSTP-RTDA-EOPG-SW-04-0008
ESIT – Enhanced Service Infrastructure Technology, Issue 1.1, March 2005

ECCS-E40B
Space Engineering Software

ESIT-SPMP-001-SPB
ESIT Software Project Management Plan

ESIT-SPAP-001-SPB
ESIT Software Product Assurance Plan

ESIT-SCMP-001-SPB
ESIT Software Configuration Management Plan

WCTS
Web Coordinate Transformation Service (WCTS) draft Implementation Specification, OGC 04-072, Version 0.2, Interoperability Program Report, 11/10/2004.

DIMAP
DIMAP format, http://www.spotimage.fr/dimap/spec/dimap.htm and http://www.spotimage.fr/dimap/spec/dictionary/Spot_Scene/@index.htm

GEOSS
IEEE-OGC-ISPRS GEOSS Workshop, "The User and the GEOSS Architecture - Applications for Asia and Pacific Rim", 24 July 2005 in Seoul, South-Korea, http://portal.opengeospatial.org/files/?artifact_id=11667 and http://portal.opengeospatial.org/files/?artifact_id=11668

WSIF
Web Services Invocation framework (WSIF), http://ws.apache.org/wsif/

BPELWS
Business Process Execution Language for Web Services, Matjaz B Juric et al., chapter 4, PACKT Publishing, http://www.packtpub.com/book/BPEL, http://www.oracle.com/technology/books/pdfs/1183_BPEL_Preview_Ch4.zip

ORTHO
Using a three dimensional spatial database to orthorectify automatically remote sensing images, S. Baillarin et al., http://www.ipi.uni-hannover.de/html/aktuelles/earsel-workshop/Baillarin.pdf

OWS2
OWS 2 Image Handling for Decision Support: Service Chaining with BPEL, 04-078, Version 1.0, Interoperability Program Report, 05/10/2004.

WNS
Web Notification Service, 03-008r2, Version 0.1.0, Discussion Paper, 21/04/2003.

GMU WCTS
An introduction to implementing GMU WCTS, http://laits.gmu.edu/webservice/WCTS/GMUWCTS.htm

BPEL
BPEL Cookbook Part 2: Building a Web services network with BPEL, http://www.oracle.com/technology/pub/articles/bpel_cookbook/index.html

A. Annex: BPEL Workflows

8.1 Demo GEOSS Korea

8.1.1 WSDL

8.1.1.1 PCI WCTS

```
<?xml version="1.0"?>
<definitions xmlns:tns="http://www.opengis.net/wcts"
xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/"
xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" targetNamespace="http://www.opengis.net/wcts"
name="WCTSPostService">
    <types>
        <schema attributeFormDefault="qualified" elementFormDefault="qualified"
targetNamespace="http://www.opengis.net/wcts" xmlns="http://www.w3.org/2001/XMLSchema">
            <element name="TransformedData" type="anyType"/>
            <element name="ExceptionReport" type="anyType"/>
        </schema>
    </types>
    <message name="WCTSPostServiceRequestMessage">
        <part name="inputfile" type="xsd:string"/>
        <part name="dimfile" type="xsd:string"/>
        <part name="demfile" type="xsd:string"/>
        <part name="bands" type="xsd:string"/>
        <part name="pxsout" type="xsd:string"/>
        <part name="outformat" type="xsd:string"/>
        <part name="sourcecrs" type="xsd:string"/>
        <part name="targetcrs" type="xsd:string"/>
        <part name="store" type="xsd:string"/>
    </message>
    <message name="WCTSPostServiceResponseMessage">
        <part name="payload" element="tns:TransformedData"/>
    </message>
    <message name="ExceptionReportFault">
        <part name="payload" element="tns:ExceptionReport"/>
    </message>
    <portType name="PCIWCTSPostService">
        <operation name="Transform">
            <input message="tns:WCTSPostServiceRequestMessage"/>
            <output message="tns:WCTSPostServiceResponseMessage"/>
            <!--fault name="exception" message="tns:ExceptionReportFault"/-->
        </operation>
    </portType>
    <binding name="PCIWCTSPost" type="tns:PCIWCTSPostService">
        <http:binding verb="POST"/>
        <operation name="Transform">
            <http:operation location="/runspotortho"/>
            <input>
                <http:urlEncoded/>
                <mime:content type="application/x-www-form-urlencoded"/>
```

```
                </input>
                <output>
                    <mime:mimeXml part="payload"/>
                    <mime:content type="text/xml"/>
                </output>
                <!--fault name="exception"/-->
            </operation>
        </binding>
        <service name="PCIWCTSPostService">
            <port name="PCIWCTSPost" binding="tns:PCIWCTSPost">
                <!--http:address location="http://mass.spacebel.be:8080/wcts"/-->
                <http:address location="http://gws2.pcigeomatics.com/wcts"/>
            </port>
        </service>
        <plnk:partnerLinkType name="WCTSPostService">
            <plnk:role name="WCTSPostServiceProvider">
                <plnk:portType name="tns:PCIWCTSPostService"/>
            </plnk:role>
        </plnk:partnerLinkType>
    </definitions>
```

8.1.1.2 Spot Image WCTS

```
<?xml version="1.0"?>
<definitions xmlns:tns="http://www.opengis.net/wcts"
xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/"
xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
targetNamespace="http://www.opengis.net/wcts" name="WCTSPostService">
    <types>
        <schema attributeFormDefault="qualified" elementFormDefault="qualified"
targetNamespace="http://www.opengis.net/wcts" xmlns="http://www.w3.org/2001/XMLSchema">
            <element name="Transform" type="anyType"/>
            <element name="TransformedData" type="anyType"/>
        </schema>
    </types>
    <message name="WCTSPostServiceRequestMessage">
        <part name="payload" element="tns:Transform"/>
    </message>
    <message name="WCTSPostServiceResponseMessage">
        <part name="payload" element="tns:TransformedData"/>
    </message>
    <portType name="WCTSPostService">
        <operation name="Transform">
            <input message="tns:WCTSPostServiceRequestMessage"/>
            <output message="tns:WCTSPostServiceResponseMessage"/>
        </operation>
    </portType>
    <binding name="WCTSPost" type="tns:WCTSPostService">
        <http:binding verb="POST"/>
```

```
                <http:operation location=""/>
                <input>
                    <mime:mimeXml part="payload"/>
                    <mime:content type="text/xml"/>
                </input>
                <output>
                    <mime:mimeXml part="payload"/>
                    <mime:content type="text/xml"/>
                </output>
            </operation>
        </binding>
        <service name="WCTSPostService">
            <port name="WCTSPost" binding="tns:WCTSPost">
                <!--http:address location="http://localhost:9001/sisa_ws_d/wcts"/-->
                <http:address location="http://ws.spotimage.com/sisa_ws_d/wcts"/>
            </port>
        </service>
        <plnk:partnerLinkType name="WCTSPostService">
            <plnk:role name="WCTSPostServiceProvider">
                <plnk:portType name="tns:WCTSPostService"/>
            </plnk:role>
        </plnk:partnerLinkType>
    </definitions>
```

8.1.2 BPEL

8.1.2.1 PCI WCTS

```
<!-- WCTS BPEL Process [Generated by the Oracle BPEL Designer] -->
<process name="WCTS" targetNamespace="http://www.opengis.net/wfdl/wcts"
suppressJoinFailure="yes" xmlns:tns="http://www.opengis.net/wfdl/wcts"
xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:bpelx="http://schemas.oracle.com/bpel/extension"
xmlns:ora="http://schemas.oracle.com/xpath/extension" xmlns:nsxml0="http://www.opengis.net/wcts"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:nsxml1="http://www.esa.int/mass"
xmlns:nsxml2="http://www.opengis.net/gml" xmlns:nsxml3="http://earth.esa.int/XML/eoli"
xmlns:nsxml4="http://www.esa.int/xml/schemas/mass/aoifeatures"
xmlns:nsxml5="http://www.esa.int/oi"
xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/">
    <!-- ================================================================= -->
    <!-- PARTNERLINKS                                                      -->
```

```
    <!-- ================================================================== -->
    <partnerLinks>
        <!-- The 'client' role represents the requester of this service. -->
        <partnerLink name="client" partnerLinkType="tns:WCTS" myRole="WCTSProvider"/>
        <partnerLink name="remotePCIWCTS" partnerLinkType="nsxml0:WCTSPostService"
partnerRole="WCTSPostServiceProvider"/>
    </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="input" messageType="tns:WCTSRequestMessage"/>
        <!--
      Reference to the message that will be returned to the requester
      -->
        <variable name="output" messageType="tns:WCTSResponseMessage"/>
        <variable name="wctsRequest"
messageType="nsxml0:WCTSPostServiceRequestMessage"/>
        <variable name="wctsResponse"
messageType="nsxml0:WCTSPostServiceResponseMessage"/>
        <variable name="imageId" type="xsd:string"/>
        <variable name="format" type="xsd:string"/>
        <variable name="width" type="xsd:string"/>
        <variable name="height" type="xsd:string"/>
        <variable name="crs" type="xsd:string"/>
        <variable name="band" type="xsd:string"/>
        <!-- for wcts-->
        <variable name="inputFile" type="xsd:string"/>
        <variable name="dimFile" type="xsd:string"/>
        <variable name="demFile" type="xsd:string"/>
        <variable name="bands" type="xsd:string"/>
        <variable name="pxsout" type="xsd:string"/>
        <variable name="outformat" type="xsd:string"/>
        <variable name="sourcecrs" type="xsd:string"/>
        <variable name="targetcrs" type="xsd:string"/>
        <variable name="store" type="xsd:string"/>
    </variables>
    <!-- ================================================================== -->
    <!-- ORCHESTRATION LOGIC                                                -->
    <!-- Set of activities coordinating the flow of messages across the    -->
    <!-- services integrated within this business process                  -->
    <!-- ================================================================== -->
    <faultHandlers>
        <catchAll>
            <sequence>
                <assign name="prepareErrorReport">
                    <copy>
                        <from>
                            <processOrderOutputMsg xmlns="http://www.esa.int/mass">
                                <getOrderOutput>
                                    <statusInfo>
                                        <statusId>300</statusId>
                                        <statusMsg>TDB</statusMsg>
```

```
                                                </getOrderOutput>
                                            </processOrderOutputMsg>
                                        </from>
                                        <to variable="output" part="parameters"
query="/nsxml1:processOrderOutputMsg"/>
                                    </copy>
                                    <copy>
                                        <from expression="concat(&quot;Order processing failed. System error
occurred in workflow Instance # &quot;, ora:getInstanceId())"/>
                                        <to variable="output" part="parameters"
query="/nsxml1:processOrderOutputMsg/nsxml1:getOrderOutput/nsxml1:statusInfo/nsxml1:statusMs
g"/>
                                    </copy>
                                </assign>
                                <reply name="reportError" partnerLink="client" portType="tns:WCTS"
operation="main" variable="output"/>
```

```
                            </sequence>
```

```
                    </catchAll>
                </faultHandlers>
                <sequence name="main">
                    <!-- Receive input from requester.
                     Note: This maps to operation defined in WCTS.wsdl
                     -->
                    <receive name="receiveInput" partnerLink="client" portType="tns:WCTS" operation="main"
variable="input" createInstance="yes"/>
                    <!-- Generate reply to synchronous request -->
                    <assign name="extractInputParam">
                        <copy>
                            <from variable="input" part="parameters"
query="/nsxml1:processOrderInputMsg/nsxml1:sendOrderInput/nsxml1:imageId"/>
                            <to variable="imageId"/>
                        </copy>
                        <copy>
                            <from variable="input" part="parameters"
query="/nsxml1:processOrderInputMsg/nsxml1:sendOrderInput/nsxml1:format"/>
                            <to variable="format"/>
                        </copy>
                        <copy>
                            <from variable="input" part="parameters"
query="/nsxml1:processOrderInputMsg/nsxml1:sendOrderInput/nsxml1:width"/>
                            <to variable="width"/>
                        </copy>
                        <copy>
                            <from variable="input" part="parameters"
query="/nsxml1:processOrderInputMsg/nsxml1:sendOrderInput/nsxml1:height"/>
                            <to variable="height"/>
                        </copy>
                        <copy>
                            <from variable="input" part="parameters"
query="/nsxml1:processOrderInputMsg/nsxml1:sendOrderInput/nsxml1:crs"/>
                            <to variable="crs"/>
                        </copy>
                        <copy>
```



```
                    <to variable="band"/>
                </copy>
                <copy>
                    <from variable="input" part="parameters"
query="/nsxml1:processOrderInputMsg/nsxml1:sendOrderInput/nsxml1:bands"/>
                    <to variable="bands"/>
                </copy>
                <copy>
                    <from variable="input" part="parameters"
query="/nsxml1:processOrderInputMsg/nsxml1:sendOrderInput/nsxml1:pxsout"/>
                    <to variable="pxsout"/>
                </copy>
                <copy>
                    <from variable="input" part="parameters"
query="/nsxml1:processOrderInputMsg/nsxml1:sendOrderInput/nsxml1:outformat"/>
                    <to variable="outformat"/>
                </copy>
                <copy>
                    <from variable="input" part="parameters"
query="/nsxml1:processOrderInputMsg/nsxml1:sendOrderInput/nsxml1:sourcecrs"/>
                    <to variable="sourcecrs"/>
                </copy>
                <copy>
                    <from variable="input" part="parameters"
query="/nsxml1:processOrderInputMsg/nsxml1:sendOrderInput/nsxml1:targetcrs"/>
                    <to variable="targetcrs"/>
                </copy>
                <copy>
                    <from variable="input" part="parameters"
query="/nsxml1:processOrderInputMsg/nsxml1:sendOrderInput/nsxml1:store"/>
                    <to variable="store"/>
                </copy>
            </assign>
            <assign name="setInputFile">
                <copy>
                    <from
expression="concat(&quot;http://ws.spotimage.com/sisa_wcs_sd/coverage/&quot;,
bpws:getVariableData(&quot;imageId&quot;),
&quot;?request=GetCoverage&amp;service=WCS&amp;version=1.0.20&amp;COVERAGE=TIF&am
p;store=false&amp;CRS=Image&amp;RESPONSE_CRS=image&amp;BBOX=0,0,6000,6000&amp;
WIDTH=&quot;, bpws:getVariableData(&quot;width&quot;), &quot;&amp;HEIGHT=&quot;,
bpws:getVariableData(&quot;height&quot;), &quot;&amp;measurename=&quot;,
bpws:getVariableData(&quot;band&quot;), &quot;&amp;FORMAT=&quot;,
bpws:getVariableData(&quot;format&quot;),
&quot;&amp;EXCEPTIONS=application/vnd.ogc.se_xml&quot;)"/>
                    <to variable="inputFile"/>
                </copy>
            </assign>
            <assign name="setDimFile">
                <copy>
                    <from
expression="concat(&quot;http://ws.spotimage.fr/sisa_wcs_sd/coverage/&quot;,
bpws:getVariableData(&quot;imageId&quot;),
```



```
&quot;/REQUEST/getdir/DIR/metadata/DATA/LPR/&quot;,
bpws:getVariableData(&quot;imageId&quot;), &quot;/metadata.dim&quot;)"/>
                <to variable="dimFile"/>
            </copy>
        </assign>
        <assign name="setDemFile">
            <copy>
                <from
expression="concat(&quot;http://ws.spotimage.com/sisa_wcs_sd/coverage/DEMPOOL?request=Get
Coverage&amp;service=WCS&amp;version=1.0.20&amp;COVERAGE=TIF&amp;store=false&amp;
CRS=&quot;, bpws:getVariableData(&quot;sourcecrs&quot;),
&quot;&amp;RESPONSE_CRS=&quot;,
bpws:getVariableData(&quot;sourcecrs&quot;),&quot;&amp;BBOX=-118,32,-
116,34&amp;WIDTH=500&amp;HEIGHT=500&amp;measurename=band1&amp;FORMAT=GeoTIFF
&amp;EXCEPTIONS=application/vnd.ogc.se_xml&quot;)"/>
                <to variable="demFile"/>
            </copy>
        </assign>
        <assign name="assignInputFileValue">
            <copy>
                <from variable="inputFile"/>
                <to variable="wctsRequest" part="inputfile"/>
            </copy>
        </assign>
        <assign name="assignDimFileValue">
            <copy>
                <from variable="dimFile"/>
                <to variable="wctsRequest" part="dimfile"/>
            </copy>
        </assign>
        <assign name="assignDemFileValue">
            <copy>
                <from variable="demFile"/>
                <to variable="wctsRequest" part="demfile"/>
            </copy>
        </assign>
        <assign name="assignBandsValue">
            <copy>
                <from variable="bands"/>
                <to variable="wctsRequest" part="bands"/>
            </copy>
        </assign>
        <assign name="pxsoutValue">
            <copy>
                <from variable="pxsout"/>
                <to variable="wctsRequest" part="pxsout"/>
            </copy>
        </assign>
        <assign name="outformatValue">
            <copy>
                <from variable="outformat"/>
                <to variable="wctsRequest" part="outformat"/>
            </copy>
        </assign>
        <assign name="sourceCRS">
```

```
                <from variable="sourcecrs"/>
                <to variable="wctsRequest" part="sourcecrs"/>
            </copy>
        </assign>
        <assign name="targetCRS">
            <copy>
                <from variable="targetcrs"/>
                <to variable="wctsRequest" part="targetcrs"/>
            </copy>
        </assign>
        <assign name="store">
            <copy>
                <from variable="store"/>
                <to variable="wctsRequest" part="store"/>
            </copy>
        </assign>
        <invoke name="callRemoteTransform" partnerLink="remotePCIWCTS"
portType="nsxml0:PCIWCTSPostService" operation="Transform" inputVariable="wctsRequest"
outputVariable="wctsResponse"/>
        <assign name="setOutput">
            <copy>
                <from expression="ora:processXSLT(&quot;xslt/transform2.xsl&quot;,
bpws:getVariableData(&quot;wctsResponse&quot;, &quot;payload&quot;,
&quot;/nsxml0:TransformedData&quot;))"/>
                <to variable="output" part="parameters"
query="/nsxml1:processOrderOutputMsg"/>
            </copy>
        </assign>
        <reply name="replyOutput" partnerLink="client" portType="tns:WCTS" operation="main"
variable="output"/>
    </sequence>
</process>
```

8.1.2.2 Spot Image WCTS

```
<!-- WCTS BPEL Process [Generated by the Oracle BPEL Designer] -->
<process name="WCTS" targetNamespace="http://www.opengis.net/wfdl/wcts"
suppressJoinFailure="yes" xmlns:tns="http://www.opengis.net/wfdl/wcts"
xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:bpelx="http://schemas.oracle.com/bpel/extension"
xmlns:ora="http://schemas.oracle.com/xpath/extension" xmlns:nsxml0="http://www.opengis.net/wcts"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:nsxml1="http://www.esa.int/mass"
xmlns:nsxml2="http://www.opengis.net/gml" xmlns:nsxml3="http://earth.esa.int/XML/eoli"
xmlns:nsxml4="http://www.esa.int/xml/schemas/mass/aoifeatures"
xmlns:nsxml5="http://www.esa.int/oi"
xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/">
    <!-- ================================================================= -->
    <!-- PARTNERLINKS                                                      -->
    <!-- List of services participating in this BPEL process               -->
    <!-- ================================================================= -->
    <partnerLinks>
        <!-- The 'client' role represents the requester of this service. -->
        <partnerLink name="client" partnerLinkType="tns:WCTS" myRole="WCTSProvider"/>
        <partnerLink name="remoteWCTS" partnerLinkType="nsxml0:WCTSPostService"
partnerRole="WCTSPostServiceProvider"/>
```

```
<!-- ================================================================== -->
<!-- VARIABLES                                                  -->
<!-- List of messages and XML documents used within this BPEL process  -->
<!-- ================================================================== -->
<variables>
    <!-- Reference to the message passed as input during initiation -->
    <variable name="input" messageType="tns:WCTSRequestMessage"/>
    <!--
   Reference to the message that will be returned to the requester
   -->
    <variable name="output" messageType="tns:WCTSResponseMessage"/>
    <variable name="wctsRequest"
messageType="nsxml0:WCTSPostServiceRequestMessage"/>
    <variable name="wctsResponse"
messageType="nsxml0:WCTSPostServiceResponseMessage"/>
    <variable name="dimUrl" type="xsd:string"/>
    <variable name="refLink" type="xsd:string"/>
    <variable name="tempInput" element="nsxml1:wctsTempInput"/>
    <variable name="imageId" type="xsd:string"/>
    <variable name="wcsHostUrl" type="xsd:string"/>
    <variable name="format" type="xsd:string"/>
    <variable name="width" type="xsd:string"/>
    <variable name="height" type="xsd:string"/>
    <variable name="crs" type="xsd:string"/>
    <variable name="band" type="xsd:string"/>
    <variable name="outputFormat" type="xsd:string"/>
</variables>
<!-- ================================================================== -->
<!-- ORCHESTRATION LOGIC                                        -->
<!-- Set of activities coordinating the flow of messages across the    -->
<!-- services integrated within this business process          -->
<!-- ================================================================== -->
<faultHandlers>
    <catchAll>
        <sequence>
            <assign name="prepareErrorReport">
                <copy>
                    <from>
                        <processOrderOutputMsg xmlns="http://www.esa.int/mass">
                            <getOrderOutput>
                                <statusInfo>
                                    <statusId>300</statusId>
                                    <statusMsg>TDB</statusMsg>
                                </statusInfo>
                            </getOrderOutput>
                        </processOrderOutputMsg>
                    </from>
                    <to variable="output" part="parameters"
query="/nsxml1:processOrderOutputMsg"/>
                </copy>
                <copy>
                    <from expression="concat(&quot;Order processing failed. System error
occurred in workflow Instance # &quot;, ora:getInstanceId())"/>
```

```
                        </copy>
                    </assign>
                    <reply name="reportError" partnerLink="client" portType="tns:WCTS"
operation="main" variable="output"/>
                </sequence>
            </catchAll>
    </faultHandlers>
    <sequence name="main">
            <!-- Receive input from requester.
            Note: This maps to operation defined in WCTS.wsdl
            -->
            <receive name="receiveInput" partnerLink="client" portType="tns:WCTS" operation="main"
variable="input" createInstance="yes"/>
            <!-- Generate reply to synchronous request -->
            <assign name="extractInputParam">
                <copy>
                    <from variable="input" part="parameters"
query="/nsxml1:processOrderInputMsg/nsxml1:sendOrderInput/nsxml1:imageId"/>
                    <to variable="imageId"/>
                </copy>
                <copy>
                    <from variable="input" part="parameters"
query="/nsxml1:processOrderInputMsg/nsxml1:sendOrderInput/nsxml1:format"/>
                    <to variable="format"/>
                </copy>
                <copy>
                    <from variable="input" part="parameters"
query="/nsxml1:processOrderInputMsg/nsxml1:sendOrderInput/nsxml1:width"/>
                    <to variable="width"/>
                </copy>
                <copy>
                    <from variable="input" part="parameters"
query="/nsxml1:processOrderInputMsg/nsxml1:sendOrderInput/nsxml1:height"/>
                    <to variable="height"/>
                </copy>
                <copy>
                    <from variable="input" part="parameters"
query="/nsxml1:processOrderInputMsg/nsxml1:sendOrderInput/nsxml1:crs"/>
                    <to variable="crs"/>
                </copy>
                <copy>
                    <from variable="input" part="parameters"
query="/nsxml1:processOrderInputMsg/nsxml1:sendOrderInput/nsxml1:band"/>
                    <to variable="band"/>
                </copy>
                <copy>
                    <from variable="input" part="parameters"
query="/nsxml1:processOrderInputMsg/nsxml1:sendOrderInput/nsxml1:outputFormat"/>
                    <to variable="outputFormat"/>
                </copy>
            </assign>
            <switch name="switch-1">
```

```
                        <assign>
                              <copy>
                                    <from
expression="&quot;http://ws.spotimage.com/sisa_wcs_sd/coverage/&quot;"/>
                                    <to variable="wcsHostUrl"/>
                              </copy>
                        </assign>
                  </case>
                  <case condition="starts-with(bpws:getVariableData(&quot;imageId&quot;),
&quot;52&quot;)">
                        <assign>
                              <copy>
                                    <from expression="&quot;http://ws.spotimage.fr/wcs/coverage/&quot;"/>
                                    <to variable="wcsHostUrl"/>
                              </copy>
                        </assign>
                  </case>
                  <otherwise>
                        <assign>
                              <copy>
                                    <from
expression="&quot;http://ws.spotimage.com/sisa_wcs_sd/coverage/&quot;"/>
                                    <to variable="wcsHostUrl"/>
                              </copy>
                        </assign>
                  </otherwise>
            </switch>
            <assign name="setValueFileParam">
                  <copy>
                        <from expression="concat(bpws:getVariableData(&quot;wcsHostUrl&quot;),
bpws:getVariableData(&quot;imageId&quot;),
&quot;/REQUEST/getdir/DIR/metadata/DATA/LPR/&quot;,
bpws:getVariableData(&quot;imageId&quot;), &quot;/metadata.dim&quot;)"/>
                        <to variable="dimUrl"/>
                  </copy>
            </assign>
            <assign name="setRefLinkParam">
                  <copy>
                        <from expression="concat(bpws:getVariableData(&quot;wcsHostUrl&quot;),
bpws:getVariableData(&quot;imageId&quot;),&quot;?request=GetCoverage&amp;service=WCS&am
p;version=1.0.20&amp;COVERAGE=&quot;,bpws:getVariableData(&quot;imageId&quot;),&quot;&am
p;store=false&amp;CRS=&quot;,bpws:getVariableData(&quot;crs&quot;),&quot;&amp;RESPONSE_
CRS=image&amp;BBOX=0,0,6000,6000&amp;WIDTH=&quot;,
bpws:getVariableData(&quot;width&quot;),&quot;&amp;HEIGHT=&quot;,bpws:getVariableData(&quo
t;height&quot;),&quot;&amp;measurename=&quot;,
bpws:getVariableData(&quot;band&quot;),&quot;&amp;FORMAT=&quot;,
bpws:getVariableData(&quot;format&quot;),
&quot;&amp;EXCEPTIONS=application/vnd.ogc.se_xml&quot;)"/>
                        <to variable="refLink"/>
                  </copy>
            </assign>
            <assign name="prepareTempInput">
                  <copy>
```

```
                <to variable="tempInput" query="/nsxml1:wctsTempInput/nsxml1:dimFileUrl"/>
            </copy>
            <copy>
                <from variable="refLink"/>
                <to variable="tempInput" query="/nsxml1:wctsTempInput/nsxml1:linkRef"/>
            </copy>
            <copy>
                <from variable="outputFormat"/>
                <to variable="tempInput" query="/nsxml1:wctsTempInput/nsxml1:outputFormat"/>
            </copy>
        </assign>
        <assign name="assignRequestValue">
            <copy>
                <from expression="ora:processXSLT(&quot;xslt/transform1.xsl&quot;,
bpws:getVariableData(&quot;tempInput&quot;, &quot;/nsxml1:wctsTempInput&quot;))"/>
                <to variable="wctsRequest" part="payload" query="/nsxml0:Transform"/>
            </copy>
        </assign>
        <invoke name="callRemoteTransform" partnerLink="remoteWCTS"
portType="nsxml0:WCTSPostService" operation="Transform" inputVariable="wctsRequest"
outputVariable="wctsResponse"/>
        <assign name="setOutput">
            <copy>
                <from expression="ora:processXSLT(&quot;xslt/transform2.xsl&quot;,
bpws:getVariableData(&quot;wctsResponse&quot;, &quot;payload&quot;,
&quot;/nsxml0:TransformedData&quot;))"/>
                <to variable="output" part="parameters"
query="/nsxml1:processOrderOutputMsg"/>
            </copy>
        </assign>
        <reply name="replyOutput" partnerLink="client" portType="tns:WCTS" operation="main"
variable="output"/>
    </sequence>
</process>
```

8.2 Demo OWS-3

8.2.1 WSDL

8.2.1.1 Spot Image “wcts.wsdl”

The current section defines the asynchronous WCTS interface.

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

```
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:fo="http://www.w3.org/1999/XSL/Format" xmlns:mass="http://www.esa.int/mass"
xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/"
xmlns:tns="http://www.opengis.net/wcts"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
targetNamespace="http://www.opengis.net/wcts">
   <types>
         <schema attributeFormDefault="qualified" elementFormDefault="qualified"
targetNamespace="http://schemas.xmlsoap.org/wsdl/"
xmlns="http://www.w3.org/2001/XMLSchema">
               <import namespace="http://www.opengis.net/wcts" schemaLocation="wcts.xsd"/>
               <import namespace="http://schemas.xmlsoap.org/ws/2003/03/addressing"
schemaLocation="http://services.eoportal.org/schemas/1.1/ws-addressing.xsd"/>
         </schema>
   </types>
   <message name="StartHeader">
         <part name="MessageID" element="wsa:MessageID"/>
         <part name="ReplyTo" element="wsa:ReplyTo"/>
   </message>
   <message name="ContinueHeader">
         <part name="RelatesTo" element="wsa:RelatesTo"/>
   </message>
   <message name="sendOrderInput">
         <part name="parameters" element="tns:sendOrderInputMsg"/>
   </message>
   <message name="sendOrderOutput">
         <part name="parameters" element="tns:sendOrderOutputMsg"/>
   </message>
   <message name="returnOrderResultInput">
         <part name="parameters" element="tns:returnOrderResultInputMsg"/>
   </message>
   <message name="returnOrderResultOutput">
         <part name="parameters" element="tns:returnOrderResultOutputMsg"/>
   </message>
   <portType name="WCTS">
         <operation name="sendOrder">
               <input name="sendOrderInput" message="tns:sendOrderInput"/>
         </operation>
   </portType>
   <portType name="WCTSCallback">
         <operation name="returnOrderResult">
               <input name="returnOrderResultInput" message="tns:returnOrderResultInput"/>
         </operation>
   </portType>
   <binding name="WCTSSoapBinding" type="tns:WCTS">
         <wsdlsoap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
         <operation name="sendOrder">
               <wsdlsoap:operation soapAction="sendOrder"/>
               <input>
                     <wsdlsoap:header message="tns:StartHeader" part="MessageID" use="literal"/>
                     <wsdlsoap:header message="tns:StartHeader" part="ReplyTo" use="literal"/>
                     <wsdlsoap:body use="literal"/>
               </input>
         </operation>
   </binding>
   <binding name="WCTSCallbackSoapBinding" type="tns:WCTSCallback">
         <wsdlsoap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
```

```
                <wsdlsoap:operation soapAction="returnOrderResult"/>
                <input>
            <operation name="returnOrderResult">
```

```
                    <wsdlsoap:header message="tns:ContinueHeader" part="RelatesTo"
use="literal"/>
```

```
                    <wsdlsoap:body use="literal"/>
                </input>
            </operation>
        </binding>
        <service name="WCTS">
            <port name="WCTS" binding="tns:WCTSSoapBinding">
                <wsdlsoap:address location="http://ws.spotimage.fr/sisa_ws/wcts"/>
            </port>
        </service>
        <service name="WCTSCallback">
            <port name="WCTSCallback" binding="tns:WCTSCallbackSoapBinding">
                <wsdlsoap:address location="value will set at runtime by means of ws-addressing"/>
            </port>
        </service>
        <plnk:partnerLinkType name="WCTS">
            <plnk:role name="WCTSServiceProvider">
                <plnk:portType name="tns:WCTS"/>
            </plnk:role>
            <plnk:role name="WCTSServiceRequester">
                <plnk:portType name="tns:WCTSCallback"/>
            </plnk:role>
        </plnk:partnerLinkType>
</definitions>
```

8.2.1.2 Spot Image “wcts.xsd”

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="http://www.opengis.net/wcts" elementFormDefault="qualified"
attributeFormDefault="qualified" xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:mass="http://www.esa.int/mass" xmlns:ns1="http://www.opengis.net/wcts">
    <import namespace="http://www.esa.int/mass"
schemaLocation="http://services.eoportal.org/schemas/1.3draft/sse_common.xsd"/>
    <include schemaLocation="http://services.eoportal.org/schemas/1.3draft/ogc.xsd"/>
    <element name="sendOrderInputMsg">
        <complexType>
            <sequence>
                <element ref="mass:commonInput"/>
                <element name="sendOrderInput">
                    <complexType>
                        <sequence>
                            <element ref="ns1:Transform"/>
                        </sequence>
                    </complexType>
                </element>
            </sequence>
```

```
        </element>
        <element name="sendOrderOutputMsg">
            <complexType>
                <sequence>
                    <element ref="mass:statusInfo"/>
                </sequence>
            </complexType>
```

```
        </element>
```

```
        <element name="returnOrderResultInputMsg">
            <complexType>
                <sequence>
                    <element ref="mass:commonInput"/>
                    <element name="getOrderOutput">
                        <complexType>
                            <sequence>
                                <element ref="mass:statusInfo"/>
                                <element ref="ns1:TransformedData" minOccurs="0"/>
                            </sequence>
                        </complexType>
                    </element>
                </sequence>
            </complexType>
        </element>
        <element name="returnOrderResultOutputMsg">
            <complexType>
                <sequence>
                    <element ref="mass:statusInfo"/>
                </sequence>
            </complexType>
        </element>
    </schema>
```

8.2.1.3 Spot Image “sias.wsdl”

This interface is the asynchronous ordering interface for products made available by Spot Image to order an image using its Dali catalogue product identifier. It also returns the WCS URL where the image can be downloaded.

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:fo="http://www.w3.org/1999/XSL/Format" xmlns:mass="http://www.esa.int/mass"
xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/"
xmlns:tns="http://www.spotimage.com/sias"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing"
```

```
        xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
        targetNamespace="http://www.spotimage.com/sias">
    <types>
            <schema attributeFormDefault="qualified" elementFormDefault="qualified"
        targetNamespace="http://schemas.xmlsoap.org/wsdl/"
        xmlns="http://www.w3.org/2001/XMLSchema">
                <import namespace="http://www. spotimage.com/sias" schemaLocation="sias.xsd"/>
                <import namespace="http://schemas.xmlsoap.org/ws/2003/03/addressing"
        schemaLocation="http://services.eoportal.org/schemas/1.1/ws-addressing.xsd"/>
            </schema>
    </types>
    <message name="StartHeader">
            <part name="MessageID" element="wsa:MessageID"/>
            <part name="ReplyTo" element="wsa:ReplyTo"/>
    </message>
    <message name="ContinueHeader">
            <part name="RelatesTo" element="wsa:RelatesTo"/>
```

```
    </message>
```

```
    <message name="sendOrderInput">
            <part name="parameters" element="tns:sendOrderInputMsg"/>
    </message>
    <message name="sendOrderOutput">
            <part name="parameters" element="tns:sendOrderOutputMsg"/>
    </message>
    <message name="returnOrderResultInput">
            <part name="parameters" element="tns:returnOrderResultInputMsg"/>
    </message>
    <message name="returnOrderResultOutput">
            <part name="parameters" element="tns:returnOrderResultOutputMsg"/>
    </message>
    <portType name="SIAS">
            <operation name="sendOrder">
                <input name="sendOrderInput" message="tns:sendOrderInput"/>
            </operation>
    </portType>
    <portType name="SIASCallback">
            <operation name="returnOrderResult">
                <input name="returnOrderResultInput" message="tns:returnOrderResultInput"/>
            </operation>
    </portType>
    <binding name="SIASSoapBinding" type="tns:SIAS">
            <wsdlsoap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
            <operation name="sendOrder">
                <wsdlsoap:operation soapAction="sendOrder"/>
                <input>
                    <wsdlsoap:header message="tns:StartHeader" part="MessageID" use="literal"/>
                    <wsdlsoap:header message="tns:StartHeader" part="ReplyTo" use="literal"/>
                    <wsdlsoap:body use="literal"/>
                </input>
            </operation>
    </binding>
    <binding name="SIASCallbackSoapBinding" type="tns:SIASCallback">
            <wsdlsoap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
            <operation name="returnOrderResult">
                <wsdlsoap:operation soapAction="returnOrderResult"/>
                <input>
```

```
                <wsdlsoap:body use="literal"/>
            </input>
        </operation>
    </binding>
    <service name="SIAS">
        <port name="SIAS" binding="tns:SIASSoapBinding">
            <wsdlsoap:address location="http://ws.spotimage.fr/sisa_ws/sias"/>
            <!--wsdlsoap:address location="http://imasoft.spotimage.fr/sisa_ws/sias"/-->
        </port>
    </service>
    <service name="SIASCallback">
        <port name="SIASCallback" binding="tns:SIASCallbackSoapBinding">
            <wsdlsoap:address location="value will set at runtime by means of ws-addressing"/>
        </port>
    </service>
    <plnk:partnerLinkType name="SIAS">
        <plnk:role name="SIASServiceProvider">
            <plnk:portType name="tns:SIAS"/>
        </plnk:role>
        <plnk:role name="SIASServiceRequester">
            <plnk:portType name="tns:SIASCallback"/>
        </plnk:role>
    </plnk:partnerLinkType>
</definitions>
```

8.2.1.4 Spot Image “sias.xsd”

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v2004 rel. 4 U (http://www.xmlspy.com) by Joel Lim (Spacebel) -->
<schema targetNamespace="http://www.spotimage.com/sias" elementFormDefault="qualified"
attributeFormDefault="qualified" xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:mass="http://www.esa.int/mass">
    <import namespace="http://www.esa.int/mass"
schemaLocation="http://services.eoportal.org/schemas/1.3draft/sse_common.xsd"/>
    <element name="sendOrderInputMsg">
        <complexType>
            <sequence>
                <element ref="mass:commonInput"/>
                <element name="sendOrderInput">
                    <complexType>
                        <sequence>
                            <element name="productId" type="string"/>
                        </sequence>
                    </complexType>
```

```
                    </sequence>
                </complexType>
            </element>
            <element name="sendOrderOutputMsg">
                <complexType>
                    <sequence>
                        <element ref="mass:statusInfo"/>
                    </sequence>
                </complexType>
            </element>
            <element name="returnOrderResultInputMsg">
                <complexType>
                    <sequence>
                        <element ref="mass:commonInput"/>
                        <element name="getOrderOutput">
                            <complexType>
                                <sequence>
                                    <element ref="mass:statusInfo"/>
                                    <element name="productInfo" minOccurs="0">
                                        <complexType>
                                            <sequence>
                                                <element name="dimapFileURL"/>
                                                <element name="inputDataRef"/>
                                                <element name="format"/>
                                            </sequence>
                                        </complexType>
                                    </element>
                                </sequence>
                            </complexType>
                        </element>
                    </sequence>
                </complexType>
            </element>
            <element name="returnOrderResultOutputMsg">
                <complexType>
                    <sequence>
                        <element ref="mass:statusInfo"/>
                    </sequence>
                </complexType>
            </element>
        </schema>
```

8.2.1.5 SSE “wctsFlow.wsdl”

A SOAP client can invoke the workflow using the WSDL interface shown below.

```
<definitions
    name="wctsFlow2"
```

```
     xmlns="http://schemas.xmlsoap.org/wsdl/"
     xmlns:tns="http://www.spotimage.fr/bpel/wcts"
     xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing"
     xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/"
     xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    >
    <types>
        <schema attributeFormDefault="qualified" elementFormDefault="qualified"
targetNamespace="http://schemas.xmlsoap.org/wsdl/"
             xmlns="http://www.w3.org/2001/XMLSchema">
            <import namespace="http://www.spotimage.fr/bpel/wcts" schemaLocation="wctsFlow2.xsd"/>
            <import namespace="http://schemas.xmlsoap.org/ws/2003/03/addressing"
schemaLocation="http://services.eoportal.org:80/orabpel/xmllib/ws-addressing.xsd"/>
        </schema>
    </types>
    <message name="wctsFlow2OutputResponseMessage">
        <part name="parameters" element="tns:returnOrderResultOutputMsg"/>
    </message>
    <message name="StartHeader">
        <part name="MessageID" element="wsa:MessageID"/>
        <part name="ReplyTo" element="wsa:ReplyTo"/>
    </message>
    <message name="ContinueHeader">
        <part name="RelatesTo" element="wsa:RelatesTo"/>
    </message>
    <message name="wctsFlow2InputResponseMessage">
        <part name="parameters" element="tns:sendOrderOutputMsg"/>
    </message>
    <message name="wctsFlow2OutputRequestMessage">
        <part name="parameters" element="tns:returnOrderResultInputMsg"/>
    </message>
    <message name="wctsFlow2InputRequestMessage">
        <part name="parameters" element="tns:sendOrderInputMsg"/>
    </message>
    <portType name="wctsFlow2Callback">
        <operation name="onResult">
            <input message="tns:wctsFlow2OutputRequestMessage"/>
            <output message="tns:wctsFlow2OutputResponseMessage"/>
        </operation>
    </portType>
    <portType name="wctsFlow2">
        <operation name="initiate">
            <input message="tns:wctsFlow2InputRequestMessage"/>
            <output message="tns:wctsFlow2InputResponseMessage"/>
        </operation>
    </portType>
    <binding name="wctsFlow2Binding" type="tns:wctsFlow2">
        <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation name="initiate">
            <soap:operation style="document" soapAction="initiate"/>
            <input>
                <soap:header message="tns:StartHeader" part="MessageID" use="literal"
encodingStyle=""/>
                <soap:header message="tns:StartHeader" part="ReplyTo" use="literal" encodingStyle=""/>
```

```
            </input>
            <output>
               <soap:body use="literal"/>
            </output>
         </operation>
      </binding>
      <binding name="wctsFlow2CallbackBinding" type="tns:wctsFlow2Callback">
         <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
         <operation name="onResult">
            <soap:operation style="document" soapAction="onResult"/>
            <input>
               <soap:header message="tns:ContinueHeader" part="RelatesTo" use="literal"
encodingStyle=""/>
               <soap:body use="literal"/>
            </input>
            <output>
               <soap:body use="literal"/>
            </output>
         </operation>
      </binding>
      <service name="wctsFlow2">
         <port name="wctsFlow2Port" binding="tns:wctsFlow2Binding">
            <soap:address location="http://services.eoportal.org:80/orabpel/spotimage/wctsFlow2/1.0"/>
         </port>
      </service>
      <service name="wctsFlow2CallbackService">
         <port name="wctsFlow2CallbackPort" binding="tns:wctsFlow2CallbackBinding">
            <soap:address location="http://set.by.caller"/>
         </port>
      </service>
    <plnk:partnerLinkType name="wctsFlow2">
      <plnk:role name="wctsFlow2Provider">
        <plnk:portType name="tns:wctsFlow2"/>
      </plnk:role>
      <plnk:role name="wctsFlow2Requester">
        <plnk:portType name="tns:wctsFlow2Callback"/>
      </plnk:role>
    </plnk:partnerLinkType>
</definitions>
```

8.2.1.6 SSE “wctsFlow.xsd”

The schema below is imported by wcts.wsdl.

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="http://www.spotimage.fr/bpel/wcts" elementFormDefault="qualified"
attributeFormDefault="qualified" xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:mass="http://www.esa.int/mass" xmlns:ns1="http://www.opengis.net/wcts">
    <import namespace="http://www.esa.int/mass"
schemaLocation="http://services.eoportal.org/schemas/1.3draft/sse_common.xsd"/>
    <import namespace="http://www.opengis.net/wcts" schemaLocation="service/ogc.xsd"/>
    <element name="sendOrderInputMsg">
           <complexType>
                  <sequence>
```

```
                        <complexType>
                            <sequence>
                                <element ref="ns1:Transform"/>
                            </sequence>
                        </complexType>
                    </element>
                </sequence>
            </complexType>
        </element>
        <element name="sendOrderOutputMsg">
            <complexType>
                <sequence>
                    <element ref="mass:statusInfo"/>
                    <element ref="mass:commonInput" minOccurs="0"/>
                </sequence>
            </complexType>
        </element>
        <element name="returnOrderResultInputMsg">
            <complexType>
                <sequence>
                    <element ref="mass:commonInput"/>
                    <element name="getOrderOutput">
                        <complexType>
                            <sequence>
                                <element ref="mass:statusInfo"/>
                                <element ref="ns1:TransformedData" minOccurs="0"/>
                            </sequence>
                        </complexType>
                    </element>
                </sequence>
            </complexType>
        </element>
        <element name="returnOrderResultOutputMsg">
            <complexType>
                <sequence>
                    <element ref="mass:statusInfo"/>
                </sequence>
            </complexType>
        </element>
    </schema>
```

8.2.1.7 OGC WCTS Transform request schema

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="http://www.opengis.net/wcts" xmlns:wcts="http://www.opengis.net/wcts"
xmlns:ows="http://www.opengis.net/ows" xmlns:gml="http://www.opengis.net/gml"
xmlns="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" version="0.0.20"
xml:lang="en">
    <annotation>
        <appinfo>wctsTransform.xsd 2004/09/21</appinfo>
        <documentation>
```

```
            <description>This XML Schema encodes the Transform package of the UML model for
        </documentation>
    </annotation>
    <!-- ==============================================================
        includes and imports
    ============================================================== -->
    <include schemaLocation="wctsCommon.xsd"/>
    <import namespace="http://www.opengis.net/ows"
schemaLocation="../../ows/0.3.20/owsAdditions.xsd"/>
    <import namespace="http://www.opengis.net/ows"
schemaLocation="../../ows/0.3.20/referenceTypeUsingGML.xsd"/>
    <import namespace="http://www.opengis.net/gml"
schemaLocation="../../gml/3.1.20/base/feature.xsd"/>
    <import namespace="http://www.opengis.net/gml"
schemaLocation="../../gml/3.1.20/base/grids.xsd"/>
    <!-- ==============================================================
        elements and types
    ============================================================== -->
    <element name="Transform">
        <annotation>
            <documentation>Request to a WCTS to perform the Transform operation. This
operation transforms coordinates from one coordinate reference system into another, where those
coordinates are in geometric primitives in GML features. The InputData are the feature(s) and/or
feature collection(s) whose geometry elements are to be transformed. Alternately, this operation
transforms a coverage from one coordinate reference system into another. In this XML encoding, no
"request" parameter is included, since the element name specifies the specific operation.
</documentation>
        </annotation>
        <complexType>
            <complexContent>
                <extension base="wcts:RequestBaseType">
                    <sequence>
                        <choice>
                            <sequence>
                                <group ref="wcts:SourceAndTargetCRSs">
                                    <annotation>
                                        <documentation>Desired well-known SourceCRS
and TargetCRS, included when client is not specifying a specific coordinate operation.
</documentation>
                                    </annotation>
                                </group>
                            </sequence>
                            <element ref="wcts:transformation">
                                <annotation>
                                    <documentation>Desired coordinate operation, included
when client is specifying a specific coordinate operation, possibly a user-defined coordinate
operation. In this use, this element shall either:
* Reference a well-known coordinate operation, whose definition is known to the WCTS server
* Contain a URL from which that definition object can be retrieved, using GML encoding
* Contain the coordinate operation definition object, using GML encoding, and a xlink:href value
containing the URN that references this definition </documentation>
                                </annotation>
                            </element>
```

```
					<element ref="wcts:InputData"/>
					<element name="InterpolationMethod"
type="ows:InterpolationMethodType" minOccurs="0">
						<annotation>
							<documentation>Identifier of interpolation method to be used
for Coverage transformation. </documentation>
						</annotation>
					</element>
					<element name="Output" type="wcts:OutputType"/>
				</sequence>
			</extension>
		</complexContent>
	</complexType>
</element>
<!-- =========================================================== -->
<complexType name="OutputType">
	<annotation>
		<documentation>Asks for the Transform response to be encoded in a particular format.
Can also ask for the response to be stored remotely from the client at a URL, instead of being
returned in the operation response. </documentation>
	</annotation>
	<sequence>
		<element name="Format" type="ows:MimeType">
			<annotation>
				<documentation>Identifier of output format to be used for the transformed
features or coverage, allowing advanced WCTS servers to perform reformatting. The output formats
supported by a WCTS are listed in the Contents section of the Capabilities document.
</documentation>
			</annotation>
		</element>
	</sequence>
	<attribute name="store" type="boolean" use="optional" default="true">
		<annotation>
			<documentation>Specifies if the response transformed data should be stored,
remotely from the client at a network URL, instead of being returned within the operation response.
This attribute should be included only it this operation parameter is supported, as encoded in the
OperationsMetadata section of the Capabilities document. </documentation>
		</annotation>
	</attribute>
</complexType>
<!-- =========================================================== -->
<element name="InputData">
	<annotation>
		<documentation>Feature data input to a Transform operation. </documentation>
	</annotation>
	<complexType>
		<choice>
			<element ref="gml:_Feature"/>
			<element ref="wcts:Reference" maxOccurs="unbounded"/>
		</choice>
	</complexType>
</element>
<!-- =========================================================== -->
<element name="TransformedData">
```

```xml
                <documentation>Reply to a valid Transform operation request sent to a WCTS.
</documentation>
            </annotation>
            <complexType>
                <sequence>
                    <choice>
                        <element ref="gml:_Feature" maxOccurs="unbounded">
                            <annotation>
                                <documentation>Ordered sequence of the features and/or feature
collections that were transformed, in the order in which input data was included and/or referenced in
the Transform element. </documentation>
                            </annotation>
                        </element>
                        <element ref="wcts:Reference" maxOccurs="unbounded"/>
                    </choice>
                    <element ref="ows:Metadata" minOccurs="0" maxOccurs="unbounded">
                        <annotation>
                            <documentation>Optional unordered list of additional metadata about the
transformed features. For example, this metadata could contain or reference data quality metadata
for these features. </documentation>
                        </annotation>
                    </element>
                </sequence>
            </complexType>
    </element>
    <!-- ========================================================== -->
    <element name="Reference">
            <annotation>
                <documentation>The reference is used to address a local payload or a remote
resource. In case of local payload (multipart mime message), the xlink:href must start with the prefix
cid:. A remote resource is typically addressed by a URL. The reference can have a human redable
description and a link to metadata. The gml:remoteschema attribute is used to indicate the schema of
the remote resource (ex. XML application schema). If the resource is accessible remotely from a
HTTP Post using XML messaging, the RequestMessage should set the XML message that needs to
be sent. Format indicate the mime type of the referenced data. The Grid is used to indicate the Grid
Geometry of image format that does not store Georeferencing (PNG, JPEG). This is useful to convert
map produced by WMS for example.</documentation>
            </annotation>
            <complexType>
                <complexContent>
                    <extension base="ows:ReferenceType">
                        <sequence>
                            <element ref="gml:Grid" minOccurs="0"/>
                        </sequence>
                    </extension>
                </complexContent>
            </complexType>
    </element>
</schema>
```

8.2.2 BPEL

8.2.2.1 WctsFlow0

This workflow call the basic service WCTS at Spot Image. Isolating it in a separate workflow allows it being called from two different flows invoked by two types of clients.

```
<!-- wctsFlow0 BPEL Process [Generated by the Oracle BPEL Designer]  -->
<process name="wctsFlow0" targetNamespace="http://www.opengis.net/wcts"
suppressJoinFailure="yes" xmlns:tns="http://www.opengis.net/wcts"
xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:bpelx="http://schemas.oracle.com/bpel/extension"
xmlns:ora="http://schemas.oracle.com/xpath/extension" xmlns:nsxml0="http://www.esa.int/mass">
    <!-- ================================================================= -->
    <!-- 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="tns:wctsFlow0" myRole="wctsFlow0Provider"
partnerRole="wctsFlow0Requester"/>
        <partnerLink name="remoteWCTS" partnerLinkType="tns:WCTS"
partnerRole="WCTSServiceProvider" myRole="WCTSServiceRequester"/>
    </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="input" messageType="tns:wctsFlow0RequestMessage"/>
        <!-- Reference to the message that will be sent back to the
         requester during callback
         -->
        <variable name="output" messageType="tns:wctsFlow0ResponseMessage"/>
        <variable name="inputRequest" messageType="tns:sendOrderInput"/>
        <variable name="inputResponse" messageType="tns:sendOrderOutput"/>
        <variable name="resultRequest" messageType="tns:returnOrderResultInput"/>
        <variable name="resultResponse" messageType="tns:returnOrderResultOutput"/>
    </variables>
    <!-- ================================================================= -->
    <!-- ORCHESTRATION LOGIC                                               -->
    <!-- Set of activities coordinating the flow of messages across the    -->
    <!-- services integrated within this business process                  -->
    <!-- ================================================================= -->
    <faultHandlers>
        <catchAll>
            <sequence><assign name="reportError"><copy>
                            <from variable="input" part="parameters"
query="/tns:sendOrderInputMsg/nsxml0:commonInput"></from>
```

```
                                        <to variable="output" part="parameters"
query="/tns:returnOrderResultInputMsg/nsxml0:commonInput"/>
                                </copy>
                                <copy>
                                        <from expression="500"></from>
                                        <to variable="output" part="parameters"
query="/tns:returnOrderResultInputMsg/tns:getOrderOutput/nsxml0:statusInfo/nsxml0:statusId"/>
                                </copy>
                                <copy>
                                        <from expression="concat(&quot;An error occurred in the instance
&quot;, ora:getInstanceId())"></from>
                                        <to variable="output" part="parameters"
query="/tns:returnOrderResultInputMsg/tns:getOrderOutput/nsxml0:statusInfo/nsxml0:statusMsg"/>
                                </copy>
                        </assign>
                        <invoke name="endTheInstance" partnerLink="client"
portType="tns:wctsFlow0Callback" operation="onResult" inputVariable="output"/>
                </sequence>
        </catchAll>
    </faultHandlers>
    <sequence name="main">
        <!-- Receive input from requestor.
        Note: This maps to operation defined in wctsFlow0.wsdl
        -->
        <receive name="receiveInput" partnerLink="client" portType="tns:wctsFlow0"
operation="initiate" variable="input" createInstance="yes"/>
        <!-- Asynchronous callback to the requester.
        Note: the callback location and correlation id is transparently handled
        using WS-addressing.
        -->
        <assign name="setRequest"><copy>
                        <from variable="input" part="parameters"
query="/tns:sendOrderInputMsg"></from>
                        <to variable="inputRequest" part="parameters" query="/tns:sendOrderInputMsg"/>
                </copy>
        </assign>
        <invoke name="callRemoteWCTS" partnerLink="remoteWCTS" portType="tns:WCTS"
operation="sendOrder" inputVariable="inputRequest"/>
        <pick name="pick-1">
                <onMessage partnerLink="remoteWCTS" portType="tns:WCTSCallback"
operation="returnOrderResult" variable="resultRequest">
                        <sequence><assign name="setResult"><copy>
                                                <from variable="resultRequest" part="parameters"
query="/tns:returnOrderResultInputMsg"></from>
                                                <to variable="output" part="parameters"
query="/tns:returnOrderResultInputMsg"/>
                                        </copy>
                                </assign>
                        </sequence>
                </onMessage>
                <onAlarm for="P3D">
                        <assign name="setReport"><copy>
                                        <from><n:returnOrderResultInputMsg
xmlns:n="http://www.opengis.net/wcts" xmlns:mass="http://www.esa.int/mass">
                                                <mass:commonInput>
                                                        <mass:orderId>tbu</mass:orderId>
                                                </mass:commonInput>
                                                <n:getOrderOutput>
                                                        <mass:statusInfo>
```

```
                                        <mass:statusMsg>tbu</mass:statusMsg>
                                    </mass:statusInfo>
                                        <mass:statusId>500</mass:statusId>
```

```
                                </n:getOrderOutput>
```

```
                            </n:returnOrderResultInputMsg></from>
                        <to variable="output" part="parameters"
query="/tns:returnOrderResultInputMsg"/>
                    </copy>
                    <copy>
                        <from variable="input" part="parameters"
query="/tns:sendOrderInputMsg/nsxml0:commonInput/nsxml0:orderId"></from>
                        <to variable="output" part="parameters"
query="/tns:returnOrderResultInputMsg/nsxml0:commonInput/nsxml0:orderId"/>
                    </copy>
                    <copy>
                        <from expression="concat(&quot;Order workflow instance &quot;,
ora:getInstanceId(), &quot;was terminated because its lifetime is over.&quot;)"></from>
                        <to variable="output" part="parameters"
query="/tns:returnOrderResultInputMsg/tns:getOrderOutput/nsxml0:statusInfo/nsxml0:statusMsg"/>
                    </copy>
                </assign>
            </onAlarm>
        </pick>
        <invoke name="callbackClient" partnerLink="client" portType="tns:wctsFlow0Callback"
operation="onResult" inputVariable="output"/>
    </sequence>
</process>
```

8.2.2.2 WctsFlow1

This workflow is called by the SSE Portal and calls WctsFlow0.

```
<!-- wctsFlow1 BPEL Process [Generated by the Oracle BPEL Designer]  -->
<process name="wctsFlow1" targetNamespace="http://www.spotimage.fr/wcts"
suppressJoinFailure="yes" xmlns:tns="http://www.spotimage.fr/wcts"
xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:bpelx="http://schemas.oracle.com/bpel/extension"
xmlns:ora="http://schemas.oracle.com/xpath/extension" xmlns:nsxml0="http://www.esa.int/mass"
xmlns:nsxml1="http://www.opengis.net/wcts" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:nsxml2="http://schemas.xmlsoap.org/ws/2003/03/addressing"
xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/">
    <!-- ================================================================= -->
    <!-- PARTNERLINKS                                                      -->
    <!-- List of services participating in this BPEL process               -->
    <!-- ================================================================= -->
    <partnerLinks>
        <!--
      The 'client' role represents the requester of this service. It is
```

```
          with the client role are automatically set using WS-Addressing.
          -->
            <partnerLink name="client" partnerLinkType="tns:wctsFlow1" myRole="wctsFlow1Provider"
      partnerRole="wctsFlow1Requester"/>
            <partnerLink name="remoteWCTSWrapper" partnerLinkType="nsxml1:wctsFlow0"
      partnerRole="wctsFlow0Provider" myRole="wctsFlow0Requester"/>
        </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="input" messageType="tns:wctsFlow1RequestMessage"/>
            <!-- Reference to the message that will be sent back to the
             requester during callback
             -->
            <variable name="output" messageType="tns:wctsFlow1ResponseMessage"/>
            <variable name="request" messageType="nsxml1:wctsFlow0RequestMessage"/>
            <variable name="result" messageType="nsxml1:wctsFlow0ResponseMessage"/>
            <variable name="wcsImageURI" element="nsxml0:wcsImageURI"/>
            <variable name="inputImageLink" type="xsd:string"/>
            <variable name="imageId" type="xsd:string"/>
            <variable name="wcsHostUrl" type="xsd:string"/>
            <variable name="bbox" type="xsd:string"/>
            <variable name="format" type="xsd:string"/>
            <variable name="width" type="xsd:string"/>
            <variable name="height" type="xsd:string"/>
            <variable name="crs" type="xsd:string"/>
            <variable name="band" type="xsd:string"/>
        </variables>
        <!-- ================================================================= -->
        <!-- ORCHESTRATION LOGIC                                             -->
        <!-- Set of activities coordinating the flow of messages across the    -->
        <!-- services integrated within this business process                 -->
        <!-- ================================================================= -->
        <faultHandlers>
            <catchAll>
                <sequence>
                    <assign name="report">
                        <copy>
                            <from variable="input" part="parameters"
      query="/nsxml0:sendOrderInputMsg/nsxml0:commonInput"/>
                            <to variable="output" part="parameters"
      query="/nsxml0:returnOrderResultInputMsg/nsxml0:commonInput"/>
                        </copy>
                        <copy>
                            <from expression="500"/>
                            <to variable="output" part="parameters"
      query="/nsxml0:returnOrderResultInputMsg/nsxml0:getOrderOutput/nsxml0:statusInfo/nsxml0:statusI
      d"/>
                        </copy>
                        <copy>
                            <from expression="concat(&quot;A system error occurred while
      processing instance # &quot;, ora:getInstanceId())"/>
```

```
                    </copy>
                </assign>
                <invoke name="finishTheInstance" partnerLink="client"
portType="tns:wctsFlow1Callback" operation="onResult" inputVariable="output"/>
            </sequence>
        </catchAll>
    </faultHandlers>
    <sequence name="main">
        <!-- Receive input from requestor.
         Note: This maps to operation defined in wctsFlow1.wsdl
         -->
```

```
        <receive name="receiveInput" partnerLink="client" portType="tns:wctsFlow1"
operation="initiate" variable="input" createInstance="yes"/>
```

```
        <!-- Asynchronous callback to the requester.
         Note: the callback location and correlation id is transparently handled
         using WS-addressing.
         -->
        <switch name="testAllowedUser">
            <case condition="contains(&quot;josh,tnn,spotimage,yves,ycoene&quot;,
bpws:getVariableData(&quot;input&quot;, &quot;parameters&quot;,
&quot;/nsxml0:sendOrderInputMsg/nsxml0:sendOrderInput/nsxml0:userId&quot;)) =
&quot;true&quot;">
                <sequence>
                    <assign name="extractInputParams">
                        <copy>
                            <from variable="input" part="parameters"
query="/nsxml0:sendOrderInputMsg/nsxml0:sendOrderInput/nsxml0:imageId"/>
                            <to variable="imageId"/>
                        </copy>
                        <copy>
                            <from variable="input" part="parameters"
query="/nsxml0:sendOrderInputMsg/nsxml0:sendOrderInput/nsxml0:bbox"/>
                            <to variable="bbox"/>
                        </copy>
                        <copy>
                            <from variable="input" part="parameters"
query="/nsxml0:sendOrderInputMsg/nsxml0:sendOrderInput/nsxml0:format"/>
                            <to variable="format"/>
                        </copy>
                        <copy>
                            <from variable="input" part="parameters"
query="/nsxml0:sendOrderInputMsg/nsxml0:sendOrderInput/nsxml0:crs"/>
                            <to variable="crs"/>
                        </copy>
                        <copy>
                            <from variable="input" part="parameters"
query="/nsxml0:sendOrderInputMsg/nsxml0:sendOrderInput/nsxml0:band"/>
                            <to variable="band"/>
                        </copy>
                        <copy>
                            <from
expression="&quot;http://ws.spotimage.fr/sisa_wcs_sd/coverage/&quot;"></from>
```

```
                    </copy>
                    <copy>
                        <from expression="6000"></from>
                        <to variable="width"/>
                    </copy>
                    <copy>
                        <from expression="6000"></from>
                        <to variable="height"/>
                    </copy>
                </assign>
                <assign name="composeWCSLinks">
                    <copy>
                        <from
expression="&quot;http://ws.spotimage.com/sisa_wcs_sd/coverage/SS5_031105/REQUEST/getdir/D
IR/metadata/DATA/LPR/SS5_031105/METADATA.DIM&quot;"></from>
                        <to variable="wcsImageURI"
query="/nsxml0:wcsImageURI/nsxml0:dimap"/>
                    </copy>
                    <copy>
```

```
                        <from
expression="concat(bpws:getVariableData(&quot;wcsHostUrl&quot;),
bpws:getVariableData(&quot;imageId&quot;),&quot;?request=GetCoverage&amp;service=WCS&am
p;version=1.0.20&amp;COVERAGE=TIF&amp;store=false&amp;CRS=&quot;,bpws:getVariableData(
&quot;crs&quot;),&quot;&amp;RESPONSE_CRS=image&amp;BBOX=&quot;,
bpws:getVariableData(&quot;bbox&quot;), &quot;&amp;WIDTH=&quot;,
bpws:getVariableData(&quot;width&quot;),&quot;&amp;HEIGHT=&quot;,bpws:getVariableData(&quo
t;height&quot;),&quot;&amp;measurename=&quot;,
bpws:getVariableData(&quot;band&quot;),&quot;&amp;FORMAT=&quot;,
bpws:getVariableData(&quot;format&quot;),
&quot;&amp;EXCEPTIONS=application/vnd.ogc.se_xml&quot;)"></from>
```

```
                        <to variable="inputImageLink"/>
                    </copy>
                    <copy>
                        <from variable="inputImageLink"></from>
                        <to variable="wcsImageURI"
query="/nsxml0:wcsImageURI/nsxml0:image"/>
                    </copy>
                </assign>
                <assign name="setRequest">
                    <copy>
                        <from
expression="ora:processXSLT(&quot;xslt/transform1.xsl&quot;,
bpws:getVariableData(&quot;wcsImageURI&quot;))"/>
                        <to variable="request" part="parameters"
query="/nsxml1:sendOrderInputMsg"/>
                    </copy>
                    <copy>
                        <from variable="input" part="parameters"
query="/nsxml0:sendOrderInputMsg/nsxml0:commonInput/nsxml0:orderId"/>
                        <to variable="request" part="parameters"
query="/nsxml1:sendOrderInputMsg/nsxml0:commonInput/nsxml0:orderId"/>
                    </copy>
                </assign>
                <invoke name="callWCTSWraper" partnerLink="remoteWCTSWrapper"
portType="nsxml1:wctsFlow0" operation="initiate" inputVariable="request"/>
```

```
                    <switch name="switch-1">
                        <case
condition="bpws:getVariableData(&quot;result&quot;,&quot;parameters&quot;,&quot;/nsxml1:returnO
rderResultInputMsg/nsxml1:getOrderOutput/nsxml0:statusInfo/nsxml0:statusId&quot;) = 0">
                            <assign name="setOutput">
                                <copy>
                                    <from
expression="ora:processXSLT(&quot;xslt/transform2.xsl&quot;,
bpws:getVariableData(&quot;result&quot;, &quot;parameters&quot;,
&quot;/nsxml1:returnOrderResultInputMsg/nsxml1:getOrderOutput/nsxml1:TransformedData&quot;))"
/>
                                    <to variable="output" part="parameters"
query="/nsxml0:returnOrderResultInputMsg"/>
                                </copy>
                                <copy>
                                    <from variable="input" part="parameters"
query="/nsxml0:sendOrderInputMsg/nsxml0:commonInput/nsxml0:orderId"/>
                                    <to variable="output" part="parameters"
query="/nsxml0:returnOrderResultInputMsg/nsxml0:commonInput/nsxml0:orderId"/>
                                </copy>
                                <copy>
                                    <from variable="inputImageLink"/>
                                    <to variable="output" part="parameters"
query="/nsxml0:returnOrderResultInputMsg/nsxml0:getOrderOutput/nsxml0:sourceImageLink"/>
                                </copy>
                            </assign>
                        </case>
                        <otherwise>
                            <assign name="reportError">
                                <copy>
                                    <from variable="input" part="parameters"
query="/nsxml0:sendOrderInputMsg/nsxml0:commonInput"/>
                                    <to variable="output" part="parameters"
query="/nsxml0:returnOrderResultInputMsg/nsxml0:commonInput"/>
                                </copy>
                                <copy>
                                    <from variable="result" part="parameters"
query="/nsxml1:returnOrderResultInputMsg/nsxml1:getOrderOutput/nsxml0:statusInfo"/>
                                    <to variable="output" part="parameters"
query="/nsxml0:returnOrderResultInputMsg/nsxml0:getOrderOutput/nsxml0:statusInfo"/>
                                </copy>
                            </assign>
                        </otherwise>
                    </switch>
                </sequence>
            </case>
            <otherwise>
                <assign name="reportOutput">
                    <copy>
                        <from variable="input" part="parameters"
query="/nsxml0:sendOrderInputMsg/nsxml0:commonInput"></from>
```



```
                        </copy>
                        <copy>
                              <from expression="502"></from>
                              <to variable="output" part="parameters"
query="/nsxml0:returnOrderResultInputMsg/nsxml0:getOrderOutput/nsxml0:statusInfo/nsxml0:statusI
d"/>
                        </copy>
                        <copy>
                              <from expression="&quot;The service you ordered is currently under
testing. If you still want to try the service, please email the service's Provider. We apologize for this
inconvenience&quot;"></from>
                              <to variable="output" part="parameters"
query="/nsxml0:returnOrderResultInputMsg/nsxml0:getOrderOutput/nsxml0:statusInfo/nsxml0:status
Msg"/>
                        </copy>
                  </assign>
            </otherwise>
      </switch>
      <!--
      <assign name="simulateResult"><copy>
                  <from expression="ora:doc(&quot;ref/wctsOutput.xml&quot;)"></from>
                  <to variable="result" part="parameters"
query="/nsxml1:returnOrderResultInputMsg"/>
            </copy>
      </assign>
      -->
      <invoke name="callbackClient" partnerLink="client" portType="tns:wctsFlow1Callback"
operation="onResult" inputVariable="output"/>
   </sequence>
</process>
```

8.2.2.3 WctsFlow2

This workflow is to be called by an external SOAP client (e.g. an OWS-3 Decision Support client) and calls WctsFlow 0.

```
<!-- wctsFlow2 BPEL Process [Generated by the Oracle BPEL Designer]  -->
<process name="wctsFlow2" targetNamespace="http://www.spotimage.fr/bpel/wcts"
suppressJoinFailure="yes" xmlns:tns="http://www.spotimage.fr/bpel/wcts"
xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:bpelx="http://schemas.oracle.com/bpel/extension"
xmlns:ora="http://schemas.oracle.com/xpath/extension"
xmlns:nsxml0="http://www.spotimage.fr/wcts" xmlns:nsxml1="http://www.esa.int/mass"
xmlns:nsxml2="http://www.opengis.net/wcts" xmlns:nsxml3="http://www.esa.int/mass/bpel">
   <!-- ================================================================= -->
   <!-- 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="tns:wctsFlow2" myRole="wctsFlow2Provider"
partnerRole="wctsFlow2Requester"/>
            <partnerLink name="remoteWCTSWrapper" partnerLinkType="nsxml2:wctsFlow0"
partnerRole="wctsFlow0Provider" myRole="wctsFlow0Requester"/>
            <partnerLink name="OrderManagerAgent" partnerLinkType="nsxml3:SSEOrderIdGenerator"
partnerRole="SSEOrderIdGeneratorProvider"/>
    </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="input" messageType="tns:wctsFlow2InputRequestMessage"/>
        <!-- Reference to the message that will be sent back to the
         requester during callback
         -->
        <variable name="output" messageType="tns:wctsFlow2OutputRequestMessage"/>
        <variable name="request" messageType="nsxml2:wctsFlow0RequestMessage"/>
        <variable name="inputACK" messageType="tns:wctsFlow2InputResponseMessage"/>
        <variable name="outputACK" messageType="tns:wctsFlow2OutputResponseMessage"/>
        <variable name="result" messageType="nsxml2:wctsFlow0ResponseMessage"/>
        <variable name="getOrderIdRequest"
messageType="nsxml3:SSEOrderIdGeneratorRequestMessage"/>
        <variable name="getOrderIdResponse"
messageType="nsxml3:SSEOrderIdGeneratorResponseMessage"/>
    </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 wctsFlow2.wsdl
         -->
        <receive name="receiveInput" partnerLink="client" portType="tns:wctsFlow2"
operation="initiate" variable="input" createInstance="yes"/>
        <!-- Asynchronous callback to the requester.
         Note: the callback location and correlation id is transparently handled
         using WS-addressing.
         -->
        <invoke name="obtainNewOrderId" partnerLink="OrderManagerAgent"
portType="nsxml3:SSEOrderIdGenerator" operation="main" inputVariable="getOrderIdRequest"
outputVariable="getOrderIdResponse"/>
        <assign name="prepareACK"><copy>
                <from variable="getOrderIdResponse" part="parameters"
query="/nsxml3:orderId"></from>
                <to variable="inputACK" part="parameters"
query="/tns:sendOrderOutputMsg/nsxml1:commonInput/nsxml1:orderId"/>
            </copy>
            <copy>
                <from expression="0"></from>
```



```
                </copy>
            </assign>
            <reply name="ackInputRequest" partnerLink="client" portType="tns:wctsFlow2"
operation="initiate" variable="inputACK"/>
            <assign name="prepareRequest"><copy>
                    <from variable="getOrderIdResponse" part="parameters"
query="/nsxml3:orderId"></from>
                    <to variable="request" part="parameters"
query="/nsxml2:sendOrderInputMsg/nsxml1:commonInput/nsxml1:orderId"/>
                </copy>
                <copy>
                    <from variable="input" part="parameters"
query="/tns:sendOrderInputMsg/tns:sendOrderInput"></from>
                    <to variable="request" part="parameters"
query="/nsxml2:sendOrderInputMsg/nsxml2:sendOrderInput"/>
                </copy>
            </assign>

            <invoke name="processService" partnerLink="remoteWCTSWrapper"
portType="nsxml2:wctsFlow0" operation="initiate" inputVariable="request"/>
            <receive createInstance="no" name="collectResult" partnerLink="remoteWCTSWrapper"
portType="nsxml2:wctsFlow0Callback" operation="onResult" variable="result"/>
            <assign name="prepareOutput"><copy>
                    <from variable="getOrderIdResponse" part="parameters"
query="/nsxml3:orderId"></from>
                    <to variable="output" part="parameters"
query="/tns:returnOrderResultInputMsg/nsxml1:commonInput/nsxml1:orderId"/>
                </copy>
                <copy>
                    <from variable="result" part="parameters"
query="/nsxml2:returnOrderResultInputMsg/nsxml2:getOrderOutput"></from>
                    <to variable="output" part="parameters"
query="/tns:returnOrderResultInputMsg/tns:getOrderOutput"/>
                </copy>
            </assign>

            <!-- for simulating the remote service, useful in unit testing
            <assign name="simulateOutput"><copy>
                    <from expression="ora:doc(&quot;ref/output.xml&quot;)"></from>
                    <to variable="output" part="parameters" query="/tns:returnOrderResultInputMsg"/>
                </copy>

                <copy>

                    <from variable="getOrderIdResponse" part="parameters"
query="/nsxml3:orderId"></from>
                    <to variable="output" part="parameters"
query="/tns:returnOrderResultInputMsg/nsxml1:commonInput/nsxml1:orderId"/>
                </copy>
            </assign>
            -->
            <invoke name="callbackClient" partnerLink="client" portType="tns:wctsFlow2Callback"
operation="onResult" inputVariable="output" outputVariable="outputACK"/>
    </sequence>
</process>
```


8.2.2.4 SIASFlow0

```
<!-- siasFlow0 BPEL Process [Generated by the Oracle BPEL Designer]  -->
<process name="siasFlow0" targetNamespace="http://www.spotimage.com/sias"
suppressJoinFailure="yes" xmlns:tns="http://www.spotimage.com/sias"
xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:bpelx="http://schemas.oracle.com/bpel/extension"
xmlns:ora="http://schemas.oracle.com/xpath/extension"
xmlns:nsxml0="http://www.spotimage.com/sias" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:nsxml2="http://schemas.xmlsoap.org/ws/2003/03/addressing"
xmlns:nsxml1="http://www.esa.int/mass">
    <!-- ================================================================= -->
    <!-- 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="tns:siasFlow0" myRole="siasFlow0Provider"
partnerRole="siasFlow0Requester"/>
        <partnerLink name="remoteSIAS" partnerLinkType="nsxml0:SIAS"
partnerRole="SIASServiceProvider" myRole="SIASServiceRequester"/>
    </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="input" messageType="tns:siasFlow0RequestMessage"/>
        <variable name="output" messageType="tns:siasFlow0ResponseMessage"/>
        <variable name="siasInput" messageType="tns:sendOrderInput"/>
        <variable name="siasOutput" messageType="tns:returnOrderResultInput"/>
        <variable name="dimapFileURL" type="xsd:string"/>
    </variables>
    <!-- ================================================================= -->
    <!-- ORCHESTRATION LOGIC                                               -->
    <!-- Set of activities coordinating the flow of messages across the    -->
    <!-- services integrated within this business process                  -->
    <!-- ================================================================= -->

    <faultHandlers>

        <catchAll>
            <sequence><assign name="reportError"><copy>
                        <from><n:returnOrderResultInputMsg
xmlns:n="http://www.spotimage.com/sias" xmlns:mass="http://www.esa.int/mass">
                            <mass:commonInput>
                                <mass:orderId>set at runtime</mass:orderId>
                            </mass:commonInput>
```

```
                                                  <mass:statusInfo>
                                                       <mass:statusId>500</mass:statusId>
                                                  </mass:statusInfo>
                                             </n:getOrderOutput>
                                        </n:returnOrderResultInputMsg></from>
                                   <to variable="output" part="parameters"
query="/tns:returnOrderResultInputMsg"/>
                              </copy>
                              <copy>
                                   <from expression="concat(&quot;An error occurred in the instance
&quot;, ora:getInstanceId())"></from>
                                   <to variable="output" part="parameters"
query="/tns:returnOrderResultInputMsg/tns:getOrderOutput/nsxml1:statusInfo/nsxml1:statusMsg"/>
                              </copy>
                              <copy>
                                   <from variable="input" part="parameters"
query="/tns:sendOrderInputMsg/nsxml1:commonInput"></from>
                                   <to variable="output" part="parameters"
query="/tns:returnOrderResultInputMsg/nsxml1:commonInput"/>
                              </copy>
                         </assign>
                         <invoke name="endTheInstance" partnerLink="client"
portType="tns:siasFlow0Callback" operation="onResult" inputVariable="output"/>
                    </sequence>
               </catchAll>
          </faultHandlers>
          <sequence name="main">
               <!-- Receive input from requestor.
               Note: This maps to operation defined in siasFlow0.wsdl
               -->
               <receive name="receiveInput" partnerLink="client" portType="tns:siasFlow0"
operation="initiate" variable="input" createInstance="yes"/>
               <!-- Asynchronous callback to the requester.
               Note: the callback location and correlation id is transparently handled
               using WS-addressing.
               -->
               <assign name="setSIASInput"><copy>
                         <from variable="input" part="parameters"
query="/tns:sendOrderInputMsg"></from>
                         <to variable="siasInput" part="parameters" query="/tns:sendOrderInputMsg"/>
                    </copy>
               </assign>
               <invoke name="requestSIAS" partnerLink="remoteSIAS" portType="tns:SIAS"
operation="sendOrder" inputVariable="siasInput"/>
               <pick name="waitForSIASResult">
                    <onMessage partnerLink="remoteSIAS" operation="returnOrderResult"
portType="tns:SIASCallback" variable="siasOutput">
                         <assign name="processSIASOutput"><copy>
                                   <from variable="siasOutput" part="parameters"
query="/tns:returnOrderResultInputMsg"></from>
                                   <to variable="output" part="parameters"
query="/tns:returnOrderResultInputMsg"/>
                              </copy>
                         </assign>
                    </onMessage>
```

```
                        <assign name="composeError"><copy>
                                <from variable="output" part="parameters"
query="/tns:returnOrderResultInputMsg/nsxml1:commonInput"></from>
                                <to variable="output" part="parameters"
query="/tns:returnOrderResultInputMsg/nsxml1:commonInput"/>
                            </copy>
                            <copy>
                                <from expression="500"></from>
                                <to variable="output" part="parameters"
query="/tns:returnOrderResultInputMsg/tns:getOrderOutput/nsxml1:statusInfo/nsxml1:statusId"/>
                            </copy>
                            <copy>
                                <from expression="concat(&quot;The instance &quot;,
ora:getInstanceId(), &quot; has not received the callback from the remote SIAS service for 3 days
and been terminated by the PM system&quot;)"></from>
                                <to variable="output" part="parameters"
query="/tns:returnOrderResultInputMsg/tns:getOrderOutput/nsxml1:statusInfo/nsxml1:statusMsg"/>
                            </copy>
                        </assign>
                    </onAlarm>
                </pick>
                <invoke name="callbackClient" partnerLink="client" portType="tns:siasFlow0Callback"
operation="onResult" inputVariable="output"/>
    </sequence>
</process>
```

8.2.2.5 WctsChaining

The WctsChaining flow calls WctsFlow0 and SIASFlow0. It is called from the SSE Portal.

```
<!-- wctsChaining BPEL Process [Generated by the Oracle BPEL Designer]  -->
<process name="wctsChaining" targetNamespace="http://www.spotimage.fr/bpel/wctsChaining"
suppressJoinFailure="yes" xmlns:tns="http://www.spotimage.fr/bpel/wctsChaining"
xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:bpelx="http://schemas.oracle.com/bpel/extension"
xmlns:ora="http://schemas.oracle.com/xpath/extension"
xmlns:nsxml0="http://www.spotimage.com/sias" xmlns:nsxml1="http://www.opengis.net/wcts"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:nsxml2="http://www.esa.int/mass"
xmlns:nsxml3="http://earth.esa.int/XML/eoli"
xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:nsxml4="http://www.opengis.net/wcts">
    <!-- ================================================================= -->
    <!-- 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="tns:wctsChaining"
myRole="wctsChainingProvider" partnerRole="wctsChainingRequester"/>
            <partnerLink name="siasWrapper" partnerLinkType="nsxml0:siasFlow0"
partnerRole="siasFlow0Provider" myRole="siasFlow0Requester"/>
            <partnerLink name="wctsWrapper" partnerLinkType="nsxml1:wctsFlow0"
partnerRole="wctsFlow0Provider" myRole="wctsFlow0Requester"/>
    </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="input" messageType="tns:wctsChainingRequestMessage"/>
        <!-- Reference to the message that will be sent back to the
         requester during callback
         -->
        <variable name="output" messageType="tns:wctsChainingResponseMessage"/>
        <variable name="orderId" type="xsd:string"/>
        <variable name="productId" type="xsd:string"/>
        <variable name="siasInput" messageType="nsxml0:siasFlow0RequestMessage"/>
        <variable name="siasOutput" messageType="nsxml0:siasFlow0ResponseMessage"/>
        <variable name="wctsInput" messageType="nsxml1:wctsFlow0RequestMessage"/>
        <variable name="wctsOutput" messageType="nsxml1:wctsFlow0ResponseMessage"/>
        <variable name="checkStatus" type="xsd:string"/>
    </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 wctsChaining.wsdl
         -->
        <receive name="receiveInput" partnerLink="client" portType="tns:wctsChaining"
operation="initiate" variable="input" createInstance="yes"/>
        <!-- Asynchronous callback to the requester.
         Note: the callback location and correlation id is transparently handled
         using WS-addressing.
         -->
        <switch name="isUserAuthorised">
            <case condition="contains(&quot;josh,spotimage,tnn,yves,ycoene&quot;,
bpws:getVariableData(&quot;input&quot;, &quot;parameters&quot;,
&quot;/nsxml2:sendOrderInputMsg/nsxml2:sendOrderInput/nsxml2:userId&quot;)) =
&quot;true&quot;"><sequence><assign name="extractInputParams"><copy>
                        <from variable="input" part="parameters"
query="/nsxml2:sendOrderInputMsg/nsxml2:commonInput/nsxml2:orderId"></from>
                        <to variable="orderId"/>
                    </copy>
                    <copy>
                        <from variable="input" part="parameters"
query="/nsxml2:sendOrderInputMsg/nsxml2:searchOutput/nsxml2:retrievedData/nsxml2:Metadata[1]/
nsxml3:dataIdInfo/nsxml3:idCitation/nsxml3:resTitle"></from>
                        <to variable="productId"/>
                    </copy>
                </assign>
```



```
<sequence><scope name="orderingSIAS"><sequence><assign
name="prepareRequest"><copy>
        <from
expression="concat(bpws:getVariableData(&quot;orderId&quot;), &quot;_sias&quot;)"></from>
        <to variable="siasInput" part="parameters"
query="/nsxml0:sendOrderInputMsg/nsxml2:commonInput/nsxml2:orderId"/>
    </copy>
    <copy>
        <from variable="productId"></from>
        <to variable="siasInput" part="parameters"
query="/nsxml0:sendOrderInputMsg/nsxml0:sendOrderInput/nsxml0:productId"/>
    </copy>
  </assign>
  <invoke name="invokeSIASWrapper"
partnerLink="siasWrapper" portType="nsxml0:siasFlow0" operation="initiate"
inputVariable="siasInput"/>
  <receive createInstance="no" name="collectSIASOutput"
partnerLink="siasWrapper" portType="nsxml0:siasFlow0Callback" operation="onResult"
variable="siasOutput"/>
 </sequence>
</scope>
<switch name="checkSIASResult">
 <case
condition="bpws:getVariableData(&quot;siasOutput&quot;,&quot;parameters&quot;,&quot;/nsxml0:ret
urnOrderResultInputMsg/nsxml0:getOrderOutput/nsxml2:statusInfo/nsxml2:statusId&quot;) = 0">
  <scope name="orderingWCTS">
    <sequence name="parseResult">
      <assign name="prepareRequest">
        <copy>
          <from
expression="ora:processXSLT(&quot;xslt/transform1.xsl&quot;,
bpws:getVariableData(&quot;siasOutput&quot;, &quot;parameters&quot;,
&quot;/nsxml0:returnOrderResultInputMsg&quot;))"></from>
          <to variable="wctsInput" part="parameters"
query="/nsxml1:sendOrderInputMsg"/>
        </copy>
        <copy>
          <from
expression="concat(bpws:getVariableData(&quot;orderId&quot;), &quot;_wcts&quot;)"></from>
          <to variable="wctsInput" part="parameters"
query="/nsxml1:sendOrderInputMsg/nsxml2:commonInput/nsxml2:orderId"/>
        </copy>
      </assign>
      <invoke name="invokeWCTSWrapper"
partnerLink="wctsWrapper" portType="nsxml1:wctsFlow0" operation="initiate"
inputVariable="wctsInput"/>
      <receive createInstance="no"
name="collectWCTSOutput" partnerLink="wctsWrapper" portType="nsxml1:wctsFlow0Callback"
operation="onResult" variable="wctsOutput"/>
      <!--
      <assign name="simulateWCTSData"><copy>
          <from
expression="ora:doc(&quot;ref/wctsOutput.xml&quot;)"></from>
          <to variable="wctsOutput"
part="parameters" query="/nsxml1:returnOrderResultInputMsg"/>
```

```
                                    <copy>
                                        <from variable="wctsInput"
part="parameters" query="/nsxml1:sendOrderInputMsg/nsxml2:commonInput"></from>
                                        <to variable="wctsOutput"
part="parameters" query="/nsxml1:returnOrderResultInputMsg/nsxml2:commonInput"/>
                                    </copy>
                                </assign>
                -->
```

```
                                <switch name="switch-3">
```

```
                                    <case
condition="bpws:getVariableData(&quot;wctsOutput&quot;,&quot;parameters&quot;,&quot;/nsxml1:re
turnOrderResultInputMsg/nsxml1:getOrderOutput/nsxml2:statusInfo/nsxml2:statusId&quot;) =
0"><assign name="prepareChainOutput"><copy>
                                                    <from
expression="ora:processXSLT(&quot;xslt/transform2.xsl&quot;,
bpws:getVariableData(&quot;wctsOutput&quot;, &quot;parameters&quot;,
&quot;/nsxml1:returnOrderResultInputMsg/nsxml1:getOrderOutput/nsxml4:TransformedData&quot;))"
></from>
                                                    <to variable="output"
part="parameters" query="/nsxml2:returnOrderResultInputMsg"/>
                                                </copy>
                                                <copy>
                                                    <from variable="input"
part="parameters" query="/nsxml2:sendOrderInputMsg/nsxml2:commonInput"></from>
                                                    <to variable="output"
part="parameters" query="/nsxml2:returnOrderResultInputMsg/nsxml2:commonInput"/>
                                                </copy>
                                                <copy>
                                                    <from variable="siasOutput"
part="parameters"
query="/nsxml0:returnOrderResultInputMsg/nsxml0:getOrderOutput/nsxml0:productInfo/nsxml0:input
DataRef"></from>
                                                    <to variable="output"
part="parameters"
query="/nsxml2:returnOrderResultInputMsg/nsxml2:getOrderOutput/nsxml2:sourceImageLink"/>
                                                </copy>
                                            </assign>
                                    </case>
                                    <otherwise><assign
name="reportError"><copy>
                                                    <from variable="input"
part="parameters" query="/nsxml2:sendOrderInputMsg/nsxml2:commonInput"></from>
                                                    <to variable="output"
part="parameters" query="/nsxml2:returnOrderResultInputMsg/nsxml2:commonInput"/>
                                                </copy>
                                                <copy>
                                                    <from
expression="500"></from>
                                                    <to variable="output"
part="parameters"
query="/nsxml2:returnOrderResultInputMsg/nsxml2:getOrderOutput/nsxml2:statusInfo/nsxml2:statusI
d"/>
                                                </copy>
                                                <copy>
```

```
                                                            <to variable="output"
part="parameters"
query="/nsxml2:returnOrderResultInputMsg/nsxml2:getOrderOutput/nsxml2:statusInfo/nsxml2:status
Msg"/>
                                                        </copy>
                                                    </assign>
                                                </otherwise>
                                            </switch>
                                        </sequence>
                                    </scope>
                                </case>
                                <otherwise><assign name="prepareReport"><copy>
```

```
                                            <from variable="input" part="parameters"
query="/nsxml2:sendOrderInputMsg/nsxml2:commonInput"></from>
```

```
                                            <to variable="output" part="parameters"
query="/nsxml2:returnOrderResultInputMsg/nsxml2:commonInput"/>
                                        </copy>
                                        <copy>
                                            <from expression="500"></from>
                                            <to variable="output" part="parameters"
query="/nsxml2:returnOrderResultInputMsg/nsxml2:getOrderOutput/nsxml2:statusInfo/nsxml2:statusI
d"/>
                                        </copy>
                                        <copy>
                                            <from expression="concat(&quot;Instance &quot;,
ora:getInstanceId(), &quot; is stopped because the failure in ordering the SIAS
service&quot;)"></from>
                                            <to variable="output" part="parameters"
query="/nsxml2:returnOrderResultInputMsg/nsxml2:getOrderOutput/nsxml2:statusInfo/nsxml2:status
Msg"/>
                                        </copy>
                                    </assign>
                                </otherwise>
                            </switch>
                        </sequence>
                    </sequence>
                </case>
                <otherwise><assign name="reportOutput"><copy>
                            <from variable="input" part="parameters"
query="/nsxml2:sendOrderInputMsg/nsxml2:commonInput"></from>
                            <to variable="output" part="parameters"
query="/nsxml2:returnOrderResultInputMsg/nsxml2:commonInput"/>
                        </copy>
                        <copy>
                            <from expression="502"></from>
                            <to variable="output" part="parameters"
query="/nsxml2:returnOrderResultInputMsg/nsxml2:getOrderOutput/nsxml2:statusInfo/nsxml2:statusI
d"/>
                        </copy>
                        <copy>
                            <from expression="&quot;The service you ordered is under testing mode.
We will inform you as soon as the service is in operation. If you like to try the service as it is now,
```

```
please send email to the service's provider. We apologize for this inconvenience. Thank
you!&quot;"></from>
                                        <to variable="output" part="parameters"
query="/nsxml2:returnOrderResultInputMsg/nsxml2:getOrderOutput/nsxml2:statusInfo/nsxml2:status
Msg"/>
                                 </copy>
                          </assign>
                   </otherwise>
            </switch>
            <invoke name="callbackClient" partnerLink="client" portType="tns:wctsChainingCallback"
operation="onResult" inputVariable="output"/>
    </sequence>
</process>
```


8.3 SensorML

8.3.1 Dimap


```
<?xml version="1.0" encoding="utf-8"?>
<Dimap_Document xsi:noNamespaceSchemaLocation="D:\users\pmerigot\projets\DIMAP\Grammar
1.1.2\Spot_Scene\Spot_Scene.xsd" name="METADATA.DIM"
xmlns:sml="http://www.opengis.net/sensorML" xmlns:om="http://www.opengis.net/om"
xmlns:swe="http://www.opengis.net/swe" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
<Metadata_Id>
<METADATA_FORMAT version="1.1">DIMAP</METADATA_FORMAT>
<METADATA_PROFILE>SPOTSCENE_1A</METADATA_PROFILE>
</Metadata_Id>
<Dataset_Id>
<DATASET_NAME>SCENE 5 247-326/7 04/12/28 04:33:57 1 J</DATASET_NAME>
<COPYRIGHT>COPYRIGHT CNES 28 12 2004 04 H 33 MN 57 S</COPYRIGHT>
<DATASET_QL_PATH href="PREVIEW.JPG"/>
<DATASET_QL_FORMAT version="6b">JPEG</DATASET_QL_FORMAT>
<DATASET_TN_PATH href="ICON.JPG"/>
<DATASET_TN_FORMAT version="6b">JPEG</DATASET_TN_FORMAT>
</Dataset_Id>
<Coordinate_Reference_System>
<GEO_TABLES version="5.2">EPSG</GEO_TABLES>
<Horizontal_CS>
<HORIZONTAL_CS_CODE>epsg:4326</HORIZONTAL_CS_CODE>
<HORIZONTAL_CS_TYPE>GEOGRAPHIC</HORIZONTAL_CS_TYPE>
<HORIZONTAL_CS_NAME>WGS 84</HORIZONTAL_CS_NAME>
</Horizontal_CS>
</Coordinate_Reference_System>
<Raster_CS>
<RASTER_CS_TYPE>POINT</RASTER_CS_TYPE>
<PIXEL_ORIGIN>1</PIXEL_ORIGIN>
</Raster_CS>
<Geoposition/>
<Production>
<PRODUCT_TYPE>SCAI1C20N</PRODUCT_TYPE>
<PRODUCT_INFO>Spot SYSTEM SCENE level 1A</PRODUCT_INFO>
<DATASET_PRODUCER_NAME>SPOT_IMAGE</DATASET_PRODUCER_NAME>
```

```
<DATASET_PRODUCTION_DATE>2004-12-
31T11:21:20.000000</DATASET_PRODUCTION_DATE>
<Production_Facility>
<SOFTWARE_NAME>CAP_T</SOFTWARE_NAME>
<SOFTWARE_VERSION>SPOT5_V07_00P1</SOFTWARE_VERSION>
<PROCESSING_CENTER>SLS_TOULOUSE</PROCESSING_CENTER>
</Production_Facility>
</Production>
<Raster_Dimensions>
<NCOLS>6000</NCOLS>
<NROWS>6000</NROWS>
<NBANDS>4</NBANDS>
</Raster_Dimensions>
<Data_Processing>
<PROCESSING_LEVEL>1A</PROCESSING_LEVEL>
<GEOMETRIC_PROCESSING>RAW</GEOMETRIC_PROCESSING>
```



```
<RADIOMETRIC_PROCESSING>SYSTEM</RADIOMETRIC_PROCESSING>
```



```
<Processing_Options>
<SWIR_BAND_REGISTRATION_FLAG>Y</SWIR_BAND_REGISTRATION_FLAG>
<X_BANDS_REGISTRATION_FLAG>Y</X_BANDS_REGISTRATION_FLAG>
</Processing_Options>
</Data_Processing>
<Image_Interpretation>
<Spectral_Band_Info>
<BAND_DESCRIPTION>XS1</BAND_DESCRIPTION>
<PHYSICAL_UNIT>W.m-2.Sr-1.um-1</PHYSICAL_UNIT>
<PHYSICAL_BIAS>0.000000</PHYSICAL_BIAS>
<PHYSICAL_GAIN/>
<PHYSICAL_CALIBRATION_DATE>2002-01-01T03:00:00</PHYSICAL_CALIBRATION_DATE>
<BAND_INDEX>1</BAND_INDEX>
</Spectral_Band_Info>
<Spectral_Band_Info>
<BAND_DESCRIPTION>XS2</BAND_DESCRIPTION>
<PHYSICAL_UNIT>W.m-2.Sr-1.um-1</PHYSICAL_UNIT>
<PHYSICAL_BIAS>0.000000</PHYSICAL_BIAS>
<PHYSICAL_GAIN/>
<PHYSICAL_CALIBRATION_DATE></PHYSICAL_CALIBRATION_DATE>
<BAND_INDEX>2</BAND_INDEX>
</Spectral_Band_Info>
<Spectral_Band_Info>
<BAND_DESCRIPTION>XS3</BAND_DESCRIPTION>
<PHYSICAL_UNIT>W.m-2.Sr-1.um-1</PHYSICAL_UNIT>
<PHYSICAL_BIAS>0.000000</PHYSICAL_BIAS>
<PHYSICAL_GAIN/>
<PHYSICAL_CALIBRATION_DATE></PHYSICAL_CALIBRATION_DATE>
<BAND_INDEX>3</BAND_INDEX>
</Spectral_Band_Info>
<Spectral_Band_Info>
<BAND_DESCRIPTION>SWIR</BAND_DESCRIPTION>
<PHYSICAL_UNIT>W.m-2.Sr-1.um-1</PHYSICAL_UNIT>
<PHYSICAL_BIAS>0.000000</PHYSICAL_BIAS>
<PHYSICAL_GAIN/>
<PHYSICAL_CALIBRATION_DATE></PHYSICAL_CALIBRATION_DATE>
<BAND_INDEX>4</BAND_INDEX>
```

```
</Image_Interpretation>
<Data_Strip>
<Satellite_Time>
<UT_DATE>20085 16404.527992</UT_DATE>
<CLOCK_VALUE>4262615552</CLOCK_VALUE>
<CLOCK_PERIOD>3.9062407142e-03</CLOCK_PERIOD>
<BOARD_TIME>4262615552</BOARD_TIME>
<TAI_TUC>32</TAI_TUC>
</Satellite_Time>
<Ephemeris>
<SATELLITE_ALTITUDE>826917.650361</SATELLITE_ALTITUDE>
<NADIR_LAT>11.916873</NADIR_LAT>
<NADIR_LON>90.965116</NADIR_LON>
<Points>
<Point>
<Location>
<X>4.6454661745e+06</X>
<Y>1.4337345174e+05</Y>
<Z>5.4967369404e+06</Z>
</Location>
<Velocity>
<X>5.5917508560e+03</X>
<Y>-1.9095527700e+03</Y>
<Z>-4.6647906330e+03</Z>
</Velocity>
<TIME>2004-01-20T11:04:28</TIME>
</Point>
<Point>
<Location>
<X>4.8108486546e+06</X>
<Y>8.5664035155e+04</Y>
<Z>5.3541720158e+06</Z>
</Location>
<Velocity>
<X>5.4328482490e+03</X>
<Y>-1.9373205000e+03</Y>
<Z>-4.8387781290e+03</Z>
</Velocity>
<TIME>2004-01-20T11:04:58</TIME>
</Point>
</Points>
<DORIS_USED>Y</DORIS_USED>
</Ephemeris>
<Satellite_Attitudes>
<Raw_Attitudes>
<Aocs_Attitude>
<Angular_Speeds_List>
<Angular_Speeds>
<TIME>2004-12-28T04:33:23.824869</TIME>
<YAW>8.9999997274e-07</YAW>
<PITCH>-7.9999958792e-07</PITCH>
<ROLL>0.0000000000e+00</ROLL>
<OUT_OF_RANGE>N</OUT_OF_RANGE>
</Angular_Speeds>
</Angular_Speeds_List>
```

</Raw_Attitudes>
<Corrected_Attitudes>
<Corrected_Attitude>
<Angles>
<TIME>2004-01-20T11:06:43</TIME>
<YAW>1.1048898666e-03</YAW>
<PITCH>-4.0862285906e-04</PITCH>
<ROLL>-9.3330583274e-05</ROLL>
<OUT_OF_RANGE>N</OUT_OF_RANGE>
</Angles>
</Corrected_Attitude>
</Corrected_Attitudes>
</Satellite_Attitudes>
<Sensor_Configuration>
<Time_Stamp>
<LINE_PERIOD>1.5039934837e-03</LINE_PERIOD>
<SCENE_CENTER_TIME>2004-12-28T04:34:03.458863</SCENE_CENTER_TIME>
<SCENE_CENTER_LINE>3001</SCENE_CENTER_LINE>
<SCENE_CENTER_COL>3001</SCENE_CENTER_COL>
</Time_Stamp>
<Instrument_Look_Angles_List>
<Instrument_Look_Angles>
<VALIDITY_DATE/>
<BAND_INDEX>1</BAND_INDEX>
<Look_Angles_List>
<Look_Angles>
<DETECTOR_ID>1</DETECTOR_ID>
<PSI_X>-9.6563273956e-03</PSI_X>
<PSI_Y>-3.4e-01</PSI_Y>
</Look_Angles>
<Look_Angles>
<DETECTOR_ID>2</DETECTOR_ID>
<PSI_X>-9.6563491464e-03</PSI_X>
<PSI_Y>-3.39e-01</PSI_Y>
</Look_Angles>
<Look_Angles>
<DETECTOR_ID>3000</DETECTOR_ID>
<PSI_X>-9.8053009485e-03</PSI_X>
<PSI_Y>+3.4e-01</PSI_Y>
</Look_Angles>
</Look_Angles_List>
</Instrument_Look_Angles>
<Instrument_Look_Angles>
<VALIDITY_DATE/>
<BAND_INDEX>2</BAND_INDEX>
<Look_Angles_List/>
</Instrument_Look_Angles>
<Instrument_Look_Angles>
<VALIDITY_DATE/>
<BAND_INDEX>3</BAND_INDEX>
<Look_Angles_List/>
</Instrument_Look_Angles>
<Instrument_Look_Angles>
<VALIDITY_DATE/>

```
<Look_Angles_List/>
</Instrument_Look_Angles>
</Instrument_Look_Angles_List>
</Sensor_Configuration>
<Frame_Counters>
<SEGMENT_START>68</SEGMENT_START>
<SCENE_START>2988</SCENE_START>
<SEGMENT_END>5407</SEGMENT_END>
</Frame_Counters>
<Data_Strip_Coordinates>
<FIRST_PIXEL_RAW>1</FIRST_PIXEL_RAW>
<FIRST_LINE_RAW>23361</FIRST_LINE_RAW>
<FIRST_PIXEL_1B>0</FIRST_PIXEL_1B>
<FIRST_LINE_1B>0</FIRST_LINE_1B>
</Data_Strip_Coordinates>
<Models/>
<Sensor_Calibration>
<METHOD>NOINFO</METHOD>
<Calibration>
<Band_Parameters>
<BAND_INDEX>1</BAND_INDEX>
</Band_Parameters>
</Calibration>
<Spectral_Sensitivities>
<Band_Spectral_Sensitivities>
<BAND_INDEX>1</BAND_INDEX>
<FIRST_WAVELENGTH_VALUE>4.500000e-07</FIRST_WAVELENGTH_VALUE>
<WAVELENGTH_STEP>5.000000e-09</WAVELENGTH_STEP>
<CALIBRATION_DATE>2000-03-01T00:00:00.000000</CALIBRATION_DATE>
<Spectral_Sensitivity_Values>
<SPECTRAL_SENSITIVITY_VALUE>0.000000</SPECTRAL_SENSITIVITY_VALUE>
</Spectral_Sensitivity_Values>
</Band_Spectral_Sensitivities>
</Spectral_Sensitivities>
<Solar_Irradiance>
<Band_Solar_Irradiance>
<BAND_INDEX>1</BAND_INDEX>
<SOLAR_IRRADIANCE_VALUE>1858</SOLAR_IRRADIANCE_VALUE>
</Band_Solar_Irradiance>
</Solar_Irradiance>
</Sensor_Calibration>
</Data_Strip>
</Dimap_Document>
```


8.3.2 Observation Spot


```
<?xml version="1.0" encoding="UTF-8"?>
<om:CommonObservation gml:id="SPOT_SCENE_345"
            xmlns:swe="http://www.opengis.net/swe"
```

```
                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                    xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:gml="http://www.opengis.net/gml"
                    xsi:schemaLocation="http://www.opengis.net/om
../../om/1.0.30/commonObservation.xsd
                                                    http://www.opengis.net/swe
../../sweCommon/1.0.30/sweCommon.xsd">
    <!---->
    <gml:name codeSpace="urn:cnes:spot5:DATASET_NAME">SCENE 5 247-326/7 04/12/28
04:33:57 1 J</gml:name>
    <gml:name codeSpace="urn:cnes:spot5:PROCESSING_LEVEL">1A</gml:name>
    <gml:name codeSpace="urn:cnes:spot5:SWIR_BAND_REGISTRATION_FLAG">Y</gml:name>
    <gml:name codeSpace="urn:cnes:spot5:X_BANDS_REGISTRATION_FLAG">Y</gml:name>
    <!---->
    <om:eventTime>
        <gml:TimePeriod>
            <gml:beginPosition>2004-12-28T04:33:23</gml:beginPosition>
            <gml:endPosition>2004-12-28T04:34:28</gml:endPosition>
        </gml:TimePeriod>
    </om:eventTime>
    <om:eventLocation/>
    <!---->
    <om:procedure xlink:href="urn:cnes:sensor:spot5"/>
    <om:observedProperty xlink:href="urn:ogc:phenomenon:radiance"/>
    <!---->
    <om:target>
        <!--
        <swe:GeoReferencableTarget>
            <gml:boundedRegion/>
            <geoLocationModel/> Tie Points Model = Model containing info in Dataset_Frame Info
and/or GeoPosition_Points
            <geoLocationModel/> Sensor Model = Reverse Sensor GeoLocation Model
        </swe:GeoReferencableTarget>
        -->
    </om:target>
    <!---->
    <om:resultDefinition>
        <swe:DataDefinition>
            <swe:dataComponents>
                <swe:DataArray>
                    <swe:size>
                        <swe:Count>6000</swe:Count> <!-- NROWS -->
                    </swe:size>
                    <swe:component name="row">
                        <swe:DataArray>
                            <swe:size>
                                <swe:Count>6000</swe:Count> <!-- NCOLS -->
                            </swe:size>
                            <swe:component name="pixel"> <!-- used to derive NBANDS -->
                                <swe:DataGroup>
                                    <swe:component name="xs1">
                                        <swe:Quantity definition="urn:ogc:data:DN"/>
                                    </swe:component>
                                    <swe:component name="xs2">
                                        <swe:Quantity definition="urn:ogc:data:DN"/>
```

```
                                                <swe:component name="xs3">
                                                    <swe:Quantity definition="urn:ogc:data:DN"/>
                                                </swe:component>
                                                <swe:component name="swir">
                                                    <swe:Quantity definition="urn:ogc:data:DN"/>
                                                </swe:component>
                                            </swe:DataGroup>
                                        </swe:component>
                                    </swe:DataArray>
                                </swe:component>
                            </swe:DataArray>
                        </swe:dataComponents>
                        <swe:encoding>
                            <!-- This section can describe binary encoding details if needed -->
                            <!-- If using TIFF, this info is already given in the TIFF header -->
                            <swe:StandardFormat mimeType="image/tiff"/>
                        </swe:encoding>
                    </swe:DataDefinition>
    </om:resultDefinition>
    <om:result externalLink="http://www.spotimage.com/spotScene.tif"/>
    <!-- This link can also point to an embedded image in another mime
        content section if using SOAP with MIME attachment for instance-->
</om:CommonObservation>
```

8.3.3 Spot 5 Scanner SensorML Example

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 rel. 4 U (http://www.xmlspy.com) by Mérigot (SPOT IMAGE) -->
<sml:SensorML xmlns:sml="http://www.opengis.net/sensorML"
xmlns:swe="http://www.opengis.net/swe" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xlink="http://www.w3.org/1999/xlink" version="1.0">
    <sml:Sensor id="SPOT5_HRG">
        <!--~~~~~~~~~~~~~~~~~~~~~~-->
        <!--General Sensor Info-->
        <!--~~~~~~~~~~~~~~~~~~~~~~-->
        <sml:identification>
            <sml:IdentifierList>
                <sml:identifier name="short_name">
                    <sml:Term qualifier="urn:ogc:identifier:shortName">HRG</sml:Term>
                </sml:identifier>
                <sml:identifier name="long_name">
                    <sml:Term qualifier="urn:ogc:identifier:longName">High Resolution
Geometric</sml:Term>
                </sml:identifier>
                <sml:identifier name="uid">
                    <sml:Term
qualifier="urn:ogc:identifier:uid">urn:cnes:sensor:spot5:v1:001</sml:Term>
                </sml:identifier>
            </sml:IdentifierList>
```

```
            <sml:classification>
                <sml:ClassifierList>
                    <sml:classifier name="sensor_type">
                        <sml:Term qualifier="urn:ogc:classifier:sensorType">Multispectral
Scanner</sml:Term>
                    </sml:classifier>
                    <sml:classifier name="application">
                        <sml:Term qualifier="urn:ogc:classifier:application">Earth
Imaging</sml:Term>
                    </sml:classifier>
                </sml:ClassifierList>
            </sml:classification>
            <!--~~~~~~~~~~-->
            <!--Contacts-->
            <!--~~~~~~~~~~-->
            <sml:contact role="operator">
                <sml:ResponsibleParty>
                    <sml:organizationName>Spot-Image</sml:organizationName>
                    <sml:contactInfo>
                        <sml:phone>
                            <sml:voice>+33 5 62 19 42 52</sml:voice>
                        </sml:phone>
                        <sml:address>
                            <sml:deliveryPoint>5, rue des satellites, BP 4359</sml:deliveryPoint>
                            <sml:city>TOULOUSE, Cedex 4</sml:city>
                            <sml:postalCode>31030</sml:postalCode>
                            <sml:country>FRANCE</sml:country>
                        </sml:address>
                    </sml:contactInfo>
                </sml:ResponsibleParty>
            </sml:contact>
            <!--~~~~~~~~~~~~~~~~~~~~~~~~~~-->
            <!--Sensor Coordinate Frame-->
            <!--~~~~~~~~~~~~~~~~~~~~~~~~~~-->
            <sml:referenceFrame>
                <sml:LocalSpatialCRS id="SENSOR_FRAME">
                    <sml:srsName>Sensor Frame</sml:srsName>
                    <sml:usesCS xlink:href="urn:ogc:cs:xyzFrame"/>
                    <sml:usesDatum>
                        <sml:LocalSpatialDatum>
                            <sml:datumName>Scanner Sensor Spatial Datum</sml:datumName>
                            <sml:anchorPoint>
                                origin is on the bottom left nut marked with a red dot
                            </sml:anchorPoint>
                            <sml:orientation>
                                x along the long edge of the scanner frame,
                                y along the scanner sweep rotation axis,
                                z toward the top of the sensor, orthogonal to the frame.
                            </sml:orientation>
                        </sml:LocalSpatialDatum>
                    </sml:usesDatum>
                </sml:LocalSpatialCRS>
```

```
            </sml:referenceFrame>
```

```
            <sml:referenceFrame>
```

```
<sml:srsName>Sensor Local Time</sml:srsName>
<sml:usesCS xlink:href="urn:ogc:crs:localTime"/>
<sml:usesDatum>
    <sml:LocalTimeDatum>
        <sml:datumName>Scanner Sensor Temporal Datum</sml:datumName>
        <sml:origin>
            origin is time at beginning of scan
        </sml:origin>
    </sml:LocalTimeDatum>
</sml:usesDatum>
</sml:LocalTimeCRS>
</sml:referenceFrame>
<!--~~~~~~~~~~~~~~~~~-->
<!--Sensor Inputs-->
<!--~~~~~~~~~~~~~~~~~-->
<sml:inputs>
    <sml:InputList>
        <sml:input name="radiation">
            <swe:Quantity definition="urn:ogc:phenomenon:radiation"/>
        </sml:input>
        <sml:input name="scan_index">
            <swe:Count definition="urn:ogc:data:index" min="1" max="3000"/>
        </sml:input>
    </sml:InputList>
</sml:inputs>
<!--~~~~~~~~~~~~~~~~~~-->
<!--Sensor Outputs-->
<!--~~~~~~~~~~~~~~~~~~-->
<sml:outputs>
    <sml:OutputList>
        <sml:output name="scanLine">
            <swe:DataArray arraySize="3000">
                <swe:component name="pixel">
                    <swe:DataGroup>
                        <swe:component name="xs1">
                            <swe:Quantity definition="urn:ogc:data:DN"/>
                        </swe:component>
                        <swe:component name="xs2">
                            <swe:Quantity definition="urn:ogc:data:DN"/>
                        </swe:component>
                        <swe:component name="xs3">
                            <swe:Quantity definition="urn:ogc:data:DN"/>
                        </swe:component>
                        <swe:component name="swir">
                            <swe:Quantity definition="urn:ogc:data:DN"/>
                        </swe:component>
                    </swe:DataGroup>
                </swe:component>
            </swe:DataArray>
        </sml:output>
    </sml:OutputList>
</sml:outputs>
<!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
<!--Internal Processes - One for each band-->
<!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
<sml:processes>
    <sml:ProcessList>
        <!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
        <!--XS1 Band Response Characteristics-->
```

```
<!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
<sml:process name="xs1_band">
	<sml:Transducer>
		<!---->
		<sml:identification>
			<sml:IdentifierList>
				<sml:identifier>
					<sml:Term
qualifier="urn:ogc:identifier:shortName">XS1</sml:Term>
				</sml:identifier>
				<sml:identifier>
					<sml:Term
qualifier="urn:cnes:identifier:bandIndex">1</sml:Term>
				</sml:identifier>
			</sml:IdentifierList>
		</sml:identification>
		<sml:characteristics>
			<sml:PropertyList>
				<sml:property name="calibration_date">
					<swe:Time>2002-01-01T03:00:00Z</swe:Time>
				</sml:property>
			</sml:PropertyList>
		</sml:characteristics>
		<!---->
		<sml:inputs>
			<sml:InputList>
				<sml:input>
					<swe:Quantity definition="urn:ogc:phenomenon:radiance"
uom="W.m-2.Sr-1.um-1"/>
				</sml:input>
				<sml:index>
					<swe:Count definition="urn:ogc:data:index" min="1"
max="3000"/>
				</sml:index>
			</sml:InputList>
		</sml:inputs>
		<!---->
		<sml:outputs>
			<sml:OutputList>
				<sml:output>
					<swe:Quantity definition="urn:ogc:data:DN" min="0"
max="255"/>
				</sml:output>
			</sml:OutputList>
		</sml:outputs>
		<!---->
		<sml:parameters>
			<sml:ParameterList>
				<sml:steadyStateResponse>
					<sml:NormalizedCurve>
						<sml:outputGain>
							<sml:IndexedCurve arraySize="3000">
								<sml:definition>
									<sml:Coordinates>
										<sml:index>
											<swe:Count
definition="urn:ogc:data:index" min="1" max="3000"/>
										</sml:index>
```

```
                                                            <swe:Quantity
definition="urn:ogc:phenomenon:gain"/>
                                                        &lt;sml:value name="cellGain"&gt;

                                                        &lt;/sml:value&gt;

                                                    &lt;/sml:Coordinates&gt;
                                                &lt;/sml:definition&gt;
                                                <!-- index/gain value pairs from
Gain_Section/PixelParameters/Cells -->
                                                &lt;!-- NaN or 0 or -1 if dead --&gt;
                                                &lt;swe:tupleValues&gt;1,0.989781 2,0.982469
... 3000,0.995738&lt;/swe:tupleValues&gt;
                                            &lt;/sml:IndexedCurve&gt;
                                        &lt;/sml:outputGain&gt;
                                        &lt;sml:function&gt;
                                            &lt;swe:Curve arraySize="2"&gt;
                                                &lt;swe:definition&gt;
                                                    &lt;swe:Coordinates&gt;
                                                        &lt;swe:axis name="radiance"&gt;
                                                            <swe:Quantity
definition="urn:ogc:phenomenon:radiance" uom="W.m-2.Sr-1.um-1"/>
                                                        &lt;/swe:axis&gt;
                                                        <swe:axis
name="digitalNumber">
                                                            <swe:Quantity
definition="urn:ogc:data:DN"/>
                                                        &lt;/swe:axis&gt;
                                                    &lt;/swe:Coordinates&gt;
                                                &lt;/swe:definition&gt;
                                                <!-- linear response curve: only two points:
80.635063 = 255/3.162396-->
                                                &lt;swe:tupleValues&gt;0,1
80.635063,255&lt;/swe:tupleValues&gt;
                                            &lt;/swe:Curve&gt;
                                        &lt;/sml:function&gt;
                                    &lt;/sml:NormalizedCurve&gt;
                                &lt;/sml:steadyStateResponse&gt;
                                &lt;sml:frequencyResponse&gt;
                                    &lt;sml:NormalizedCurve&gt;
                                        &lt;sml:function&gt;
                                            &lt;swe:Curve arraySize="40"&gt;
                                                &lt;swe:definition&gt;
                                                    &lt;swe:Coordinates&gt;
                                                        &lt;swe:axis name="wavelength"&gt;
                                                            <swe:Quantity
definition="urn:ogc:phenomenon:wavelength" uom="nm"/>
                                                        &lt;/swe:axis&gt;
                                                        &lt;swe:axis name="gain"&gt;
                                                            <swe:Quantity
definition="urn:ogc:phenomenon:gain"/>
                                                        &lt;/swe:axis&gt;
                                                    &lt;/swe:Coordinates&gt;
                                                &lt;/swe:definition&gt;
                                                <!-- wavelength/gain value pairs from
Spectral_Sensivities/Band_Spectral_Sensivities -->
                                                &lt;swe:tupleValues&gt;450,0.0 455,0.0 460,0.0
... 490,0.1373 ... 540,1.0 ... 600,0.6117 ... 650,0.0&lt;/swe:tupleValues&gt;
```

```
</sml:function>
</sml:NormalizedCurve>
</sml:frequencyResponse>
<sml:samplePosition>
<sml:PositionCurves id="SAMPLE_POSITION"
referenceFrame="#SENSOR_FRAME">
<sml:curve name="lookAngles">
<sml:NormalizedCurve>
<sml:function>
<sml:IndexedCurve
arraySize="3000">
<sml:definition>
<sml:Coordinates>
<sml:index>
<swe:Count
definition="urn:ogc:data:index" min="1" max="3000"/>
</sml:index>
<sml:value
name="psi_x">
<swe:Quantity
definition="urn:ogc:phenomenon:rotation" uom="rad" axisCode="X"/>
</sml:value>
<sml:value
name="psi_y">
<swe:Quantity
definition="urn:ogc:phenomenon:rotation" uom="rad" axisCode="Y"/>
</sml:value>
</sml:Coordinates>
</sml:definition>
<!-- index/angle value pairs from
Instrument_Look_Angles/Look_Angles_List -->
<swe:tupleValues>1,-
9.6563273956e-03,-3.4e-01 2,-9.6563491464e-03,-3.39e-01 3000,-9.8053009485e-03,+3.4e-
01</swe:tupleValues>
</sml:IndexedCurve>
</sml:function>
</sml:NormalizedCurve>
</sml:curve>
</sml:PositionCurves>
</sml:samplePosition>
</sml:ParameterList>
</sml:parameters>
<!---->
<sml:method xlink:href="urn:ogc:process:transducer"/>
</sml:Transducer>
</sml:process>
<!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
<!--XS2 Band Response Characteristics-->
<!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
<sml:process name="xs2_band">
<sml:Transducer>
<!---->
<sml:identification>
<sml:IdentifierList>
```

```
                                        <sml:Term
qualifier="urn:ogc:identifier:shortName">XS2</sml:Term>
                                    </sml:identifier>
                                    <sml:identifier>
                                        <sml:Term
qualifier="urn:cnes:identifier:bandIndex">2</sml:Term>
                                    </sml:identifier>
                                </sml:IdentifierList>
                            </sml:identification>
                            <!---->
                            <sml:inputs>
                                <sml:InputList>
                                    <sml:input>
```

```
                                        <swe:Quantity definition="urn:ogc:phenomenon:radiance"
uom="W.m-2.Sr-1.um-1"/>
```

```
                                    </sml:input>
                                    <sml:index>
                                        <swe:Count definition="urn:ogc:data:index" min="1"
max="3000"/>
                                    </sml:index>
                                </sml:InputList>
                            </sml:inputs>
                            <!---->
                            <sml:outputs>
                                <sml:OutputList>
                                    <sml:output>
                                        <swe:Quantity definition="urn:ogc:data:DN" min="0"
max="255"/>
                                    </sml:output>
                                </sml:OutputList>
                            </sml:outputs>
                            <!---->
                            <sml:parameters>
                                <sml:ParameterList>
                                    <sml:steadyStateResponse>
                                        <sml:NormalizedCurve>
                                            <sml:outputGain>
                                                <sml:IndexedCurve arraySize="3000">
                                                    <sml:definition>
                                                        <sml:Coordinates>
                                                            <sml:index>
                                                                <swe:Count
definition="urn:ogc:data:index" min="1" max="3000"/>
                                                            </sml:index>
                                                            <sml:value name="cellGain">
                                                                <swe:Quantity
definition="urn:ogc:phenomenon:gain"/>
                                                            </sml:value>
                                                        </sml:Coordinates>
                                                    </sml:definition>
                                                    <!-- index/gain value pairs from
Gain_Section/PixelParameters/Cells -->
                                                    <!-- NaN or 0 or -1 if dead -->
                                                    <swe:tupleValues>1,0.989781 2,0.982469
... 3000,0.995738</swe:tupleValues>
```

```
                                                  </sml:outputGain>
                                                  <sml:function>
                                                       <swe:Curve arraySize="2">
                                                            <swe:definition>
                                                                 <swe:Coordinates>
                                                                      <swe:axis name="radiance">
                                                                           <swe:Quantity
definition="urn:ogc:phenomenon:radiance" uom="W.m-2.Sr-1.um-1"/>
                                                                      </swe:axis>
                                                                      <swe:axis
name="digitalNumber">
                                                                           <swe:Quantity
definition="urn:ogc:data:DN"/>
                                                                      </swe:axis>
                                                                 </swe:Coordinates>
                                                            </swe:definition>
                                                            <!-- linear response curve: only two points:
80.635063 = 255/3.162396-->
```

```
                                                            <swe:tupleValues>0,1
80.635063,255</swe:tupleValues>
```

```
                                                       </swe:Curve>
                                                  </sml:function>
                                             </sml:NormalizedCurve>
                                        </sml:steadyStateResponse>
                                        <sml:frequencyResponse>
                                             <sml:NormalizedCurve>
                                                  <sml:function>
                                                       <swe:Curve arraySize="40">
                                                            <swe:definition>
                                                                 <swe:Coordinates>
                                                                      <swe:axis name="wavelength">
                                                                           <swe:Quantity
definition="urn:ogc:phenomenon:wavelength" uom="nm"/>
                                                                      </swe:axis>
                                                                      <swe:axis name="gain">
                                                                           <swe:Quantity
definition="urn:ogc:phenomenon:gain"/>
                                                                      </swe:axis>
                                                                 </swe:Coordinates>
                                                            </swe:definition>
                                                            <!-- wavelength/gain value pairs from
Spectral_Sensivities/Band_Spectral_Sensivities -->
                                                            <swe:tupleValues/>
                                                       </swe:Curve>
                                                  </sml:function>
                                             </sml:NormalizedCurve>
                                        </sml:frequencyResponse>
                                        <sml:samplePosition xlink:href="#SAMPLE_POSITION"/>
                                   </sml:ParameterList>
                              </sml:parameters>
                              <!---->
                              <sml:method xlink:href="urn:ogc:process:transducer"/>
                         </sml:Transducer>
                    </sml:process>
```

```
<!--XS3 Band Response Characteristics-->
<!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
<sml:process name="xs3_band">
	<sml:Transducer>
		<!---->
		<sml:identification>
			<sml:IdentifierList>
				<sml:identifier>
					<sml:Term
qualifier="urn:ogc:identifier:shortName">XS3</sml:Term>
				</sml:identifier>
				<sml:identifier>
					<sml:Term
qualifier="urn:cnes:identifier:bandIndex">3</sml:Term>
				</sml:identifier>
			</sml:IdentifierList>
		</sml:identification>
		<!---->
		<sml:inputs>
			<sml:InputList>
				<sml:input>
					<swe:Quantity definition="urn:ogc:phenomenon:radiance"
uom="W.m-2.Sr-1.um-1"/>
				</sml:input>
				<sml:index>
					<swe:Count definition="urn:ogc:data:index" min="1"
max="3000"/>
				</sml:index>
			</sml:InputList>
		</sml:inputs>
		<!---->
		<sml:outputs>
			<sml:OutputList>
				<sml:output>
					<swe:Quantity definition="urn:ogc:data:DN" min="0"
max="255"/>
				</sml:output>
			</sml:OutputList>
		</sml:outputs>
		<!---->
		<sml:parameters>
			<sml:ParameterList>
				<sml:steadyStateResponse>
					<sml:NormalizedCurve>
						<sml:outputGain>
							<sml:IndexedCurve arraySize="3000">
								<sml:definition>
									<sml:Coordinates>
										<sml:index>
											<swe:Count
definition="urn:ogc:data:index" min="1" max="3000"/>
										</sml:index>
										<sml:value name="cellGain">
											<swe:Quantity
definition="urn:ogc:phenomenon:gain"/>
```

```
                                        </sml:Coordinates>
                                    </sml:definition>
                                    <!-- index/gain value pairs from
Gain_Section/PixelParameters/Cells -->
                                    <!-- NaN or 0 or -1 if dead -->
                                    <swe:tupleValues>1,0.989781 2,0.982469
... 3000,0.995738</swe:tupleValues>
                                </sml:IndexedCurve>
                            </sml:outputGain>
                            <sml:function>
                                <swe:Curve arraySize="2">
                                    <swe:definition>
                                        <swe:Coordinates>
                                            <swe:axis name="radiance">
                                                <swe:Quantity
definition="urn:ogc:phenomenon:radiance" uom="W.m-2.Sr-1.um-1"/>
                                            </swe:axis>
                                            <swe:axis
name="digitalNumber">
                                                <swe:Quantity
definition="urn:ogc:data:DN"/>
                                            </swe:axis>
                                        </swe:Coordinates>
                                    </swe:definition>
                                    <!-- linear response curve: only two points:
80.635063 = 255/3.162396-->
                                    <swe:tupleValues>0,1
80.635063,255</swe:tupleValues>
                                </swe:Curve>
                            </sml:function>
```

```
                        </sml:NormalizedCurve>
```

```
                    </sml:steadyStateResponse>
                    <sml:frequencyResponse>
                        <sml:NormalizedCurve>
                            <sml:function>
                                <swe:Curve arraySize="40">
                                    <swe:definition>
                                        <swe:Coordinates>
                                            <swe:axis name="wavelength">
                                                <swe:Quantity
definition="urn:ogc:phenomenon:wavelength" uom="nm"/>
                                            </swe:axis>
                                            <swe:axis name="gain">
                                                <swe:Quantity
definition="urn:ogc:phenomenon:gain"/>
                                            </swe:axis>
                                        </swe:Coordinates>
                                    </swe:definition>
                                    <!-- wavelength/gain value pairs from
Spectral_Sensivities/Band_Spectral_Sensivities -->
                                    <swe:tupleValues/>
                                </swe:Curve>
                            </sml:function>
                        </sml:NormalizedCurve>
```

```
                                        <sml:samplePosition xlink:href="#SAMPLE_POSITION"/>
                                    </sml:ParameterList>
                                </sml:parameters>
                                <!---->
                                <sml:method xlink:href="urn:ogc:process:transducer"/>
                            </sml:Transducer>
                        </sml:process>
                        <!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
                        <!--SWIR Band Response Characteristics-->
                        <!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
                        <sml:process name="swir_band">
                            <sml:Transducer>
                                <!---->
                                <sml:identification>
                                    <sml:IdentifierList>
                                        <sml:identifier>
                                            <sml:Term
qualifier="urn:ogc:identifier:shortName">SWIR</sml:Term>
                                        </sml:identifier>
                                        <sml:identifier>
                                            <sml:Term
qualifier="urn:cnes:identifier:bandIndex">4</sml:Term>
                                        </sml:identifier>
                                    </sml:IdentifierList>
                                </sml:identification>
                                <!---->
                                <sml:inputs>
                                    <sml:InputList>
                                        <sml:input>
                                            <swe:Quantity definition="urn:ogc:phenomenon:radiance"
uom="W.m-2.Sr-1.um-1"/>
                                        </sml:input>
                                        <sml:index>
                                            <swe:Count definition="urn:ogc:data:index" min="1"
max="3000"/>
                                        </sml:index>
                                    </sml:InputList>
                                </sml:inputs>
                                <!---->
                                <sml:outputs>
                                    <sml:OutputList>
                                        <sml:output>
                                            <swe:Quantity definition="urn:ogc:data:DN" min="0"
max="255"/>
                                        </sml:output>
                                    </sml:OutputList>
                                </sml:outputs>
                                <!---->
                                <sml:parameters>
                                    <sml:ParameterList>
                                        <sml:steadyStateResponse>
                                            <sml:NormalizedCurve>
                                                <sml:outputGain>
                                                    <sml:IndexedCurve arraySize="3000">
                                                        <sml:definition>
```

```
                                        <sml:index>
                                            <swe:Count
definition="urn:ogc:data:index" min="1" max="3000"/>
                                        </sml:index>
                                        <sml:value name="cellGain">
                                            <swe:Quantity
definition="urn:ogc:phenomenon:gain"/>
                                        </sml:value>
                                    </sml:Coordinates>
                                </sml:definition>
                                <!-- index/gain value pairs from
Gain_Section/PixelParameters/Cells -->
                                <!-- NaN or 0 or -1 if dead -->
                                <swe:tupleValues>1,0.989781 2,0.982469
... 3000,0.995738</swe:tupleValues>
                            </sml:IndexedCurve>
                        </sml:outputGain>
                        <sml:function>
                            <swe:Curve arraySize="2">
                                <swe:definition>
                                    <swe:Coordinates>
                                        <swe:axis name="radiance">
                                            <swe:Quantity
definition="urn:ogc:phenomenon:radiance" uom="W.m-2.Sr-1.um-1"/>
                                        </swe:axis>
                                        <swe:axis
name="digitalNumber">
                                            <swe:Quantity
definition="urn:ogc:data:DN"/>
                                        </swe:axis>
                                    </swe:Coordinates>
                                </swe:definition>
                                <!-- linear response curve: only two points:
80.635063 = 255/3.162396-->
                                <swe:tupleValues>0,1
80.635063,255</swe:tupleValues>
                            </swe:Curve>
                        </sml:function>
                    </sml:NormalizedCurve>
                </sml:steadyStateResponse>
                <sml:frequencyResponse>
                    <sml:NormalizedCurve>
                        <sml:function>
                            <swe:Curve arraySize="40">
                                <swe:definition>
                                    <swe:Coordinates>
                                        <swe:axis name="wavelength">
                                            <swe:Quantity
definition="urn:ogc:phenomenon:wavelength" uom="nm"/>
                                        </swe:axis>
                                        <swe:axis name="gain">
                                            <swe:Quantity
definition="urn:ogc:phenomenon:gain"/>
                                        </swe:axis>
```

```
                                                  </swe:definition>
                                                  <!-- wavelength/gain value pairs from
Spectral_Sensivities/Band_Spectral_Sensivities -->
                                                  <swe:tupleValues/>
                                              </swe:Curve>
                                          </sml:function>
                                      </sml:NormalizedCurve>
                                  </sml:frequencyResponse>
                                  <sml:samplePosition xlink:href="#SAMPLE_POSITION"/>
                              </sml:ParameterList>
                          </sml:parameters>
                          <!---->
                          <sml:method xlink:href="urn:ogc:process:transducer"/>
                      </sml:Transducer>
                  </sml:process>
              </sml:ProcessList>
          </sml:processes>
          <!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
          <!--Processes Interconnections-->
          <!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
          <sml:connections>
              <sml:ConnectionList>
                  <sml:connection>
                      <sml:Link>
                          <sml:source ref="this/inputs/radiation"/>
                          <sml:destination ref="xs1_band/inputs/input"/>
                      </sml:Link>
                  </sml:connection>
                  <sml:connection>
                      <sml:Link>
                          <sml:source ref="this/inputs/scan_index"/>
                          <sml:destination ref="xs1_band/inputs/index"/>
                      </sml:Link>
                  </sml:connection>
                  <sml:connection>
                      <sml:Link>
                          <sml:source ref="this/inputs/radiation"/>
                          <sml:destination ref="xs2_band/inputs/input"/>
                      </sml:Link>
                  </sml:connection>
                  <sml:connection>
                      <sml:Link>
                          <sml:source ref="this/inputs/scan_index"/>
                          <sml:destination ref="xs2_band/inputs/index"/>
                      </sml:Link>
                  </sml:connection>
                  <sml:connection>
                      <sml:Link>
                          <sml:source ref="this/inputs/radiation"/>
                          <sml:destination ref="xs3_band/inputs/input"/>
                      </sml:Link>
                  </sml:connection>
                  <sml:connection>
                      <sml:Link>
                          <sml:source ref="this/inputs/scan_index"/>
```

```
                  </sml:Link>
               </sml:connection>
               <sml:connection>
                  <sml:Link>
                     <sml:source ref="this/inputs/radiation"/>
                     <sml:destination ref="swir_band/inputs/input"/>
                  </sml:Link>
               </sml:connection>
               <sml:connection>
                  <sml:Link>
                     <sml:source ref="this/inputs/scan_index"/>
                     <sml:destination ref="swir_band/inputs/index"/>
                  </sml:Link>
               </sml:connection>
               <sml:connection>
                  <sml:ArrayLink>
                     <sml:destinationArray ref="this/outputs/scanLine"/>
                     <sml:indexSource ref="this/inputs/scan_index"/>
                     <sml:connection>
                        <sml:Link>
                           <sml:source ref="xs1_band/ouputs/output"/>
                           <sml:destination ref="pixel/xs1"/>
                        </sml:Link>
                     </sml:connection>
                     <sml:connection>
                        <sml:Link>
                           <sml:source ref="xs2_band/ouputs/output"/>
                           <sml:destination ref="pixel/xs2"/>
                        </sml:Link>
                     </sml:connection>
                     <sml:connection>
                        <sml:Link>
                           <sml:source ref="xs3_band/ouputs/output"/>
                           <sml:destination ref="pixel/xs3"/>
                        </sml:Link>
                     </sml:connection>
                     <sml:connection>
                        <sml:Link>
                           <sml:source ref="swir_band/ouputs/output"/>
                           <sml:destination ref="pixel/swir"/>
                        </sml:Link>
                     </sml:connection>
                  </sml:ArrayLink>
               </sml:connection>
            </sml:ConnectionList>
         </sml:connections>
      </sml:Sensor>
</sml:SensorML>
```

8.3.4 Spot 5 Satellite SensorML Example

```
<?xml version="1.0" encoding="UTF-8"?>
<sml:SensorML xmlns:sml="http://www.opengis.net/sensorML"
xmlns:swe="http://www.opengis.net/swe"
```

```
            xsi:schemaLocation="http://www.opengis.net/sensorML ../base/sensorML.xsd"
version="1.0">
    <sml:System id="SPOT5_SAT">
        <!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
        <!--Satellite Coordinate Frame-->
        <!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
        <sml:referenceFrame>
            <sml:LocalSpatialCRS id="SAT_FRAME">
                <sml:srsName>Sensor Frame</sml:srsName>
                <sml:usesCS xlink:href="urn:ogc:cs:xyzFrame"/>
                <sml:usesDatum>
                    <sml:LocalSpatialDatum>
                        <sml:datumName>SPOT5 Satellite Spatial Datum</sml:datumName>
                        <sml:anchorPoint>
                            origin is at the center of mass of the satellite
                        </sml:anchorPoint>
                        <sml:orientation>
                            x (tangage) along the solar panel arm and toward the solar panel,
                            y (roulis) perpendicular to the front face of the satellite (sweep axis),
                            z (lacet) perpendicular to x and y (points to the center of the earth in
flight).
                        </sml:orientation>
                    </sml:LocalSpatialDatum>
                </sml:usesDatum>
            </sml:LocalSpatialCRS>
        </sml:referenceFrame>
        <!--~~~~~~~~~~~~~~~~~~-->
        <!--System Outputs-->
        <!--~~~~~~~~~~~~~~~~~~-->
        <sml:outputs>
            <sml:OutputList>
                <sml:output name="HRG_scanLine">
                    <swe:DataArray arraySize="3000">
                        <swe:component name="pixel">
                            <swe:DataGroup>
                                <swe:component name="xs1">
                                    <swe:Quantity definition="urn:ogc:data:DN"/>
                                </swe:component>
                                <swe:component name="xs2">
                                    <swe:Quantity definition="urn:ogc:data:DN"/>
                                </swe:component>
                                <swe:component name="xs3">
                                    <swe:Quantity definition="urn:ogc:data:DN"/>
                                </swe:component>
                                <swe:component name="swir">
                                    <swe:Quantity definition="urn:ogc:data:DN"/>
                                </swe:component>
                            </swe:DataGroup>
                        </swe:component>
                    </swe:DataArray>
                </sml:output>
            </sml:OutputList>
        </sml:outputs>
        <!--~~~~~~~~~~~~~~~~~~~~~-->
```

```
			<!--~~~~~~~~~~~~~~~~~~~~-->
			<sml:processes>
				<sml:ProcessList>
					<sml:process name="hrg_scanner" xlink:href="./SPOT5_Scanner.xml"/>
				</sml:ProcessList>
			</sml:processes>
			<!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
			<!--Processes Interconnections-->
			<!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
			<sml:connections>
				<sml:ConnectionList>
					<sml:connection>
						<sml:Link>
							<sml:source ref="hrg_scanner/outputs/scanLine"/>
							<sml:destination ref="this/outputs/HRG_scanLine"/>
						</sml:Link>
					</sml:connection>
				</sml:ConnectionList>
			</sml:connections>
			<!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
			<!--Component Frames Position-->
			<!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
			<sml:positions>
				<sml:PositionList>
					<!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
					<!-- Position of satellite relative to ECEF frame -->
					<!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
					<sml:position name="satelliteLocation">
						<sml:GenericPosition>
							<sml:inputs>
								<sml:InputList>
									<sml:time>
										<swe:Time
	definition="urn:ogc:phenomenon:time:iso8601"/>
									</sml:time>
								</sml:InputList>
							</sml:inputs>
							<sml:outputs>
								<sml:OutputList>
									<sml:position>
										<swe:LocationData
	definition="urn:ogc:phenomenon:location" localFrame="#SAT_FRAME"
	referenceFrame="urn:ogc:crs:ecef"/>
									</sml:position>
								</sml:OutputList>
							</sml:outputs>
							<sml:parameters>
								<sml:ParameterList>
									<sml:location name="doris_location">
										<swe:NormalizedCurve>
											<swe:function>
												<swe:Curve arraySize="80"> <!-- number of
	location points-->
													<swe:definition>
														<swe:Coordinates>
															<swe:axis name="t">
```

```
                                                            <swe:Time
definition="urn:ogc:phenomenon:time:iso8601"/>
                                                        </swe:axis>
                                                        <swe:axis name="x">
                                                            <swe:Quantity
axisCode="X" uom="m"/>
                                                        </swe:axis>
                                                        <swe:axis name="y">
                                                            <swe:Quantity
axisCode="Y" uom="m"/>
                                                        </swe:axis>
                                                        <swe:axis name="z">
                                                            <swe:Quantity
axisCode="Z" uom="m"/>
                                                        </swe:axis>
                                                    </swe:Coordinates>
                                                </swe:definition>
                                                <swe:tupleValues>2004-01-
20T11:04:28.0Z,4.6454661745e+06,1.4337345174e+05,5.4967369404e+06 2004-01-
20T11:04:58.0Z,4.8108486546e+06,8.5664035155e+04,5.3541720158e+06</swe:tupleValues>
                                            </swe:Curve>
                                        </swe:function>
                                    </swe:NormalizedCurve>
                                </sml:location>
                                <sml:velocity name="doris_velocity">
                                    <swe:NormalizedCurve>
                                        <swe:function>
                                            <swe:Curve arraySize="80"> <!-- number of
velocity points-->
                                                <swe:definition>
                                                    <swe:Coordinates>
                                                        <swe:axis name="t">
                                                            <swe:Time
definition="urn:ogc:phenomenon:time:iso8601"/>
                                                        </swe:axis>
                                                        <swe:axis name="vx">
                                                            <swe:Quantity
axisCode="X" uom="m.s-1"/>
                                                        </swe:axis>
                                                        <swe:axis name="vy">
                                                            <swe:Quantity
axisCode="Y" uom="m.s-1"/>
                                                        </swe:axis>
                                                        <swe:axis name="vz">
                                                            <swe:Quantity
axisCode="Z" uom="m.s-1"/>
                                                        </swe:axis>
                                                    </swe:Coordinates>
                                                </swe:definition>
                        <!--for each point in /Ephemeris-->
                                                <swe:tupleValues>2004-01-
20T11:04:28.0Z,5.5917508560e+03,-1.9095527700e+03,-4.6647906330e+03 2004-01-
20T11:04:58.0Z,5.4328482490e+03,-1.9373205000e+03,-4.8387781290e+03</swe:tupleValues>
                                            </swe:Curve>
                                        </swe:function>
                                    </swe:NormalizedCurve>
                                </sml:velocity>
                            </sml:ParameterList>
                        </sml:parameters>
```

```
                              </sml:GenericPosition>

                                   <sml:method xlink:href="urn:ogc:process:genericPosition"/>

                         </sml:position>

                         <sml:position name="satelliteAttitude">
                              <sml:GenericPosition>
                                   <sml:inputs>
                                        <sml:InputList>
                                             <sml:time>
                                                  <swe:Time
definition="urn:ogc:phenomenon:time:iso8601"/>
                                             </sml:time>
                                        </sml:InputList>
                                   </sml:inputs>
                                   <sml:outputs>
                                        <sml:OutputList>
                                             <sml:position>
                                                  <swe:OrientationData
definition="urn:ogc:phenomenon:orientation" localFrame="#SAT_FRAME"
referenceFrame="urn:ogc:crs:ecef"/>
                                             </sml:position>
                                        </sml:OutputList>
                                   </sml:outputs>
                                   <sml:parameters>
                                        <sml:ParameterList>
                                             <sml:orientation name="corrected_attitude">
                                                  <swe:NormalizedCurve>
                                                       <swe:function>
                                                            <swe:Curve arraySize="300"> <!-- number of
orientation points-->
                                                                 <swe:definition>
                                                                      <swe:Coordinates>
                                                                           <swe:axis name="t">
                                                                                <swe:Time
definition="urn:ogc:phenomenon:time:iso8601"/>
                                                                           </swe:axis>
                                                                           <swe:axis name="yaw">
                                                                                <swe:Quantity
axisCode="Z" uom="rad"/>
                                                                           </swe:axis>
                                                                           <swe:axis name="pitch">
                                                                                <swe:Quantity
axisCode="X" uom="rad"/>
                                                                           </swe:axis>
                                                                           <swe:axis name="roll">
                                                                                <swe:Quantity
axisCode="Y" uom="rad"/>
                                                                           </swe:axis>
                                                                      </swe:Coordinates>
                                                                 </swe:definition>
                    <!--for each point in /Satellite_Attitudes/Corrected_Attitudes/Angles-->
                                                                 <swe:tupleValues>2004-01-
20T11:06:43.762445Z,1.1048898666e-03,-4.0862285906e-04,-9.3330583274e-05 2004-01-
20T11:06:43.887445Z,1.1049021162e-03,-4.0890350327e-04,-9.3342130615e-
05</swe:tupleValues>
                                                            </swe:Curve>
```

```
                                        </swe:NormalizedCurve>
                                    </sml:orientation>
                                    <sml:rotationOrder>
                                        <swe:Category>ZXY</swe:Category>
                                    </sml:rotationOrder>
```

```
                                </sml:ParameterList>
```

```
                            </sml:parameters>
                            <sml:method xlink:href="urn:ogc:process:genericPosition"/>
                        </sml:GenericPosition>
                    </sml:position>
                    <!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
                    <!-- Position of sensor relative to satellite frame -->
                    <!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
                    <sml:position>
                        <swe:Position localFrame="./SPOT5_Scanner.xml#SENSOR_FRAME"
referenceFrame="#SENSOR_FRAME">
                            <swe:location>
                                <swe:Location definition="urn:ogc:phenomenon:location">
                                    <swe:x><swe:Quantity uom="m">0.0</swe:Quantity></swe:x>
                                    <swe:y><swe:Quantity uom="m">0.0</swe:Quantity></swe:y>
                                    <swe:z><swe:Quantity uom="m">0.0</swe:Quantity></swe:z>
                                </swe:Location>
                            </swe:location>
                            <swe:orientation>
                                <swe:Orientation definition="urn:ogc:phenomenon:orientation">
                                    <swe:x><swe:Quantity
uom="deg">0.0</swe:Quantity></swe:x>
                                    <swe:y><swe:Quantity
uom="deg">0.0</swe:Quantity></swe:y>
                                    <swe:z><swe:Quantity
uom="deg">0.0</swe:Quantity></swe:z>

    <swe:order><swe:Category>XYZ</swe:Category></swe:order>
                                </swe:Orientation>
                            </swe:orientation>
                        </swe:Position>
                    </sml:position>
                </sml:PositionList>
            </sml:positions>
    </sml:System>
</sml:SensorML>
```