Heterogeneous Missions Accessibility – Testbed

Sensor Planning Systems Interface

HMA-T SPS I/F

ARCHITECTURAL DESIGN DOCUMENT

Code : HMAT-DMS-ADD-001
Issue : 1.0
Date : 06/10/2008

	Name	Function	Signature
Prepared by	Reuben Wright (DMS) Philippe Mérigot (SPOT)	Project Engineer	
Reviewed by	Ricardo Moyano (DMS)		
Approved by	Ricardo Moyano (DMS)	Project Manager	
Signatures and approvals on original			

DEIMOS Space S.L.
Ronda de Poniente, 19, Edificio Fiteni VI, 2-2ª
28760 Tres Cantos (Madrid), SPAIN
Tel.: +34 91 806 34 50 / Fax: +34 91 806 34 51
E-mail: deimos@deimos-space.com

This page intentionally left blank

Document Information

Contract Data	
Contract Number:	780-2/C09 "Heterogeneous Missions Accessibility - Testbed. Phase 2"
Contract Issuer:	SPACEBEL

Internal Distribution		
Name	Unit	Copies
Ricardo Moyano	MOS	1
Reuben Wright	MOS	1
Internal Confidentiality Level (DMS-COV-POL05)		
Unclassified ☐	Restricted ☑	Confidential ☐

External Distribution		
Name	Organisation	Copies
Yves Coene	SPACEBEL	1
Pier Giorgio Marchetti	ESA/ESRIN	1

Archiving	
Word Processor:	MS Word 2000
File Name:	HMAT-DMS-ADD-001.doc

Document Status Log

Issue	Change description	Date	Approved
1.0	First version of the document	06/10/2008	

Table of Contents

List of Tables

List of Figures

1. INTRODUCTION

This project concerns the HMA-T project relating to unified methods of accessing multiple, heterogeneous missions. In particular, these work packages cover the implementation of the SPS Profile for Earth Observation interface (OGC 07-018) for two different sensor systems.

1.1. Purpose

The objective of this document is to define the software architecture of the Sensor Planning System interfaces which provide a method of accessing internal planning systems.

The intended readerships for this document are, apart from SPACEBEL and ESA, the HMA-T SPS interface developers, and the validation team.

1.2. Scope

This document details the design of the interface components, and their interaction with each other, the system clients and the internal systems.

- Section 1 provides the purpose, scope and this document's overview.
- Section 2 provides the list of reference documents.
- Section 3 provides an overview of the SPS design.
- Section 4 provides the detailed system design at the level of individual components.
- Section 5 provides traceability matrices for tracking the system requirements.

1.3. Acronyms and Abbreviations

The acronyms and abbreviations used in this document are the following ones:

Acronym	Description
AD	Architectural Design
COTS	Commercial Off-The-Shelf
DB	Data Base
DMS	DEIMOS Space
ECSS	European Cooperation on Space Standardization
EO	Earth Observation
ESA	European Space Agency
FTP	File Transfer Protocol
GMES	Global Monitoring for Environment and Security
HMA	Heterogeneous Missions Accessibility

Acronym	Description
HMA-I	HMA Interoperability
HMA-T	HMA Testbed
HW	Hardware
ICD	Interface Control Document
N/A	Not Applicable
OGC	Open Geospatial Consortium Inc.
SAR	Synthetic Aperture Radar
SOAP	Simple Object Access Protocol
SoW	Statement of Work
SPOT	SPOT Image
SRD	Software Requirements Document
SVV	Software Verification and Validation
SW	Software
TBC	To Be Confirmed
TBD	To Be Defined
UML	Unified Modelling Language
XML	Extensible Markup Language
XSD	XML Schema Definition
V&V	Verification & Validation

2. RELATED DOCUMENTS

2.1. Applicable Documents

The following table specifies the applicable documents that shall be complied with during project development.

Table 1: Applicable documents

Reference	Code	Title	Issue
[OGC 07-018]	OGC 07-018	Opengis Sensor Planning Service Application Profile for EO Sensors	draft 1.0
[SOW]	SPB-HMA-T-SOW-002	Statement of Work For HMA Testbed (HMA-T) Phase 2	1.0
[SRD]	HMAT-DMS-SRD-001	HMA-T SPS I/F System Requirements Document	1.0

2.2. Reference Documents

The following table specifies the reference documents that shall be taken into account during project development.

Table 2: Reference documents

Reference	Code	Title	Issue
[RD UML]	ISBN 0-201-57168-4	The Unified Modelling Language User Guide, Grady Booch, James Rumbaugh, Ivar Jacobson.	-
[RD PMP]	HMAT-SPMP-0002-SPB	HMA-T Phase 2 - Software Project Management Plan	1.2
[RD Notifications]	WS-BaseNotification	Web Services Base Notification OASIS Standard, 1 October 2006	1.3
[OGC 05-008]	OGC 05-008	OpenGIS Web Services Common Specification	1.0

2.3. Standards

The following table specifies the standards that shall be complied with during project development.

Table 3: Standards

Reference	Code	Title	Issue
[ECSS-40-1B]	ECSS-E-40 Part 1B	Software – Part 1: Principles and requirements	1B

Reference	Code	Title	Issue
[ECSS-40-2B]	ECSS-E-40 Part 2B	Software — Part 2: Document Requirements Definitions (DRDs)	2B

3. DESIGN OVERVIEW

The Interface is between a remote Client system and a local planning system. The former will be accessing via internet connections, whereas the latter will be accessible via normal program interfaces. The connection to the Client is via HTTP connections, primarily carrying SOAP messages.

Communication with the two specific SPS systems subject to the project shall be carried out using the mechanisms that each of them has established for the access. This means that the SPS I/F components need to translate the incoming requests coming from clients into specific commands that each SPS is able to understand and manage; the converse needs to be done when forwarding the responses provided by each SPS.

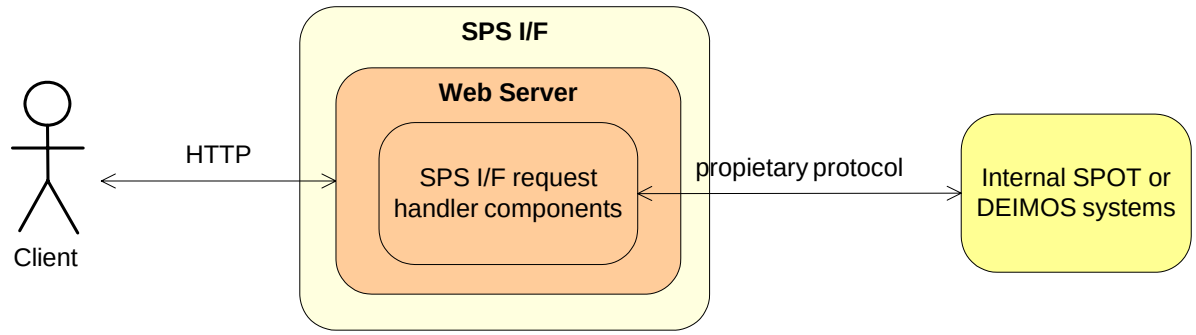

Figure 1: SPS I/F overall architecture

The operations belonging to this interface are aimed at determining the service metadata (or Capabilities) documents that describe the abilities of the specific server implementation (`GetCapabilities`), at requesting the information that is needed in order to send `GetFeasibility` and `Submit` requests (`DescribeTasking`), at determining the feasibility of an intended sensor planning request (`GetFeasibility`), for submitting such a request (`Submit`), for inquiring about the status of such a request (`GetStatus`), and for requesting information about further OGC Web services that provide access to the data collected by the requested task (`DescribeResultAccess`).

Except `GetCapabilities` which implements HTTP GET transfer using keyword-value pair (KVP) encoding (see [OGC 05-008]), all operations implement HTTP POST with XML encoding based on SOAP. All operations, including `GetFeasibility` and `Submit`, are synchronous.

In the scope of the DEIMOS development a simulated Ground Segment planning system will be used as the local system, based on COTS used in many Earth Observation mission ground segments. This will allow `GetFeasibility` responses to contain the result of the processing.

For the SPOT system a library that performs feasibility study, still under development, will be used. This library does not take the satellite's workload into account. However, these feasibility studies and satellite tasking may take days or months, so there is a need for getting the result asynchronously. To avoid polling methods the concept of notifications can be used. The scenario in this case is given below:

1. The client sends a `GetFeasibility` request to the SPS server
2. The SPS server starts the feasibility study process

3. The SPS server sends the `GetFeasibility` response to the client (feasibility study process still running)
4. The client sends a *TASK COMPLETED* subscription request to the notification server
5. Once the task is completed, the SPS server sends a notify request to the notification server
6. The notification sever sends a notify request to the client
7. The client sends a `GetStatus` request to the server to get the feasibility result

4. SYSTEM DESIGN

4.1. Design Method

The architecture and design will use UML as a key way to capture design decisions. UML is chosen due to the fact that it is general-purpose, broadly applicable, tool-supported and industry standardized modelling language. Object oriented methods enhance the simplicity on the design of complex systems. Therefore, object oriented designs tend to reduce the potential errors in producing the design documentation. UML has been used as the notation to capture the design components of the subsystem.

The design is being done in an incremental way. The initial design here defines the main SW components, and will be refined through development. This initial design also defines the internal interfaces between SW components.

This incremental approach has the main following tasks:

- Split the SW components in new components (this step has to be repeated as necessary) or will define their classes with their relationships
- Detail more deeply the static and the dynamic view of the components
- Define the tests

The UML diagrams and their uses are:

- Class and Component diagrams for describing the static view (class, methods, attributes and relationships)
- Sequence diagrams for detailing the flows between components and inside a component
- Use Case diagrams for examining shared functionality
- State diagrams for describing for the active SW components

4.2. System Decomposition

The HMA-T SPS Interface implementation consists of six parts. There is a **Web Server**, which passes SOAP messages to a **SOAP Reader/Writer** which uses a shared **SPS Library**. These produce Java objects manipulated by the **SPS Controller** which is the core of the system. It uses a **Database** for storage and produces responses which are serialized by the **SOAP Reader/Writer** (again using the **SPS Library**) and sent back to the Client through the **Web Server**. For the complex task of establishing the feasibility of a request the **SPS Controller** passes the information to a **Ground Segment Planning System**. This system will either

1. produce a response immediately and inform the **SPS Controller**, which will store the response in the **Database** and send the response to the client (DEIMOS SPS), or
2. produce a response asynchronously and inform the **SPS Controller**, which will store the response in the **Database** and trigger the **Notification Server** to contact the Client. The SPS controller communicates with the Notification server by sending *Notify* SOAP messages (cf. [RD Notifications]). The response data will be pulled out of the **Database** to provide a synchronous response to a separate Client operation. (SPOT SPS).

The web communications are handled over HTTP.

The following diagram shows these components (and some of the classes) and their interactions.

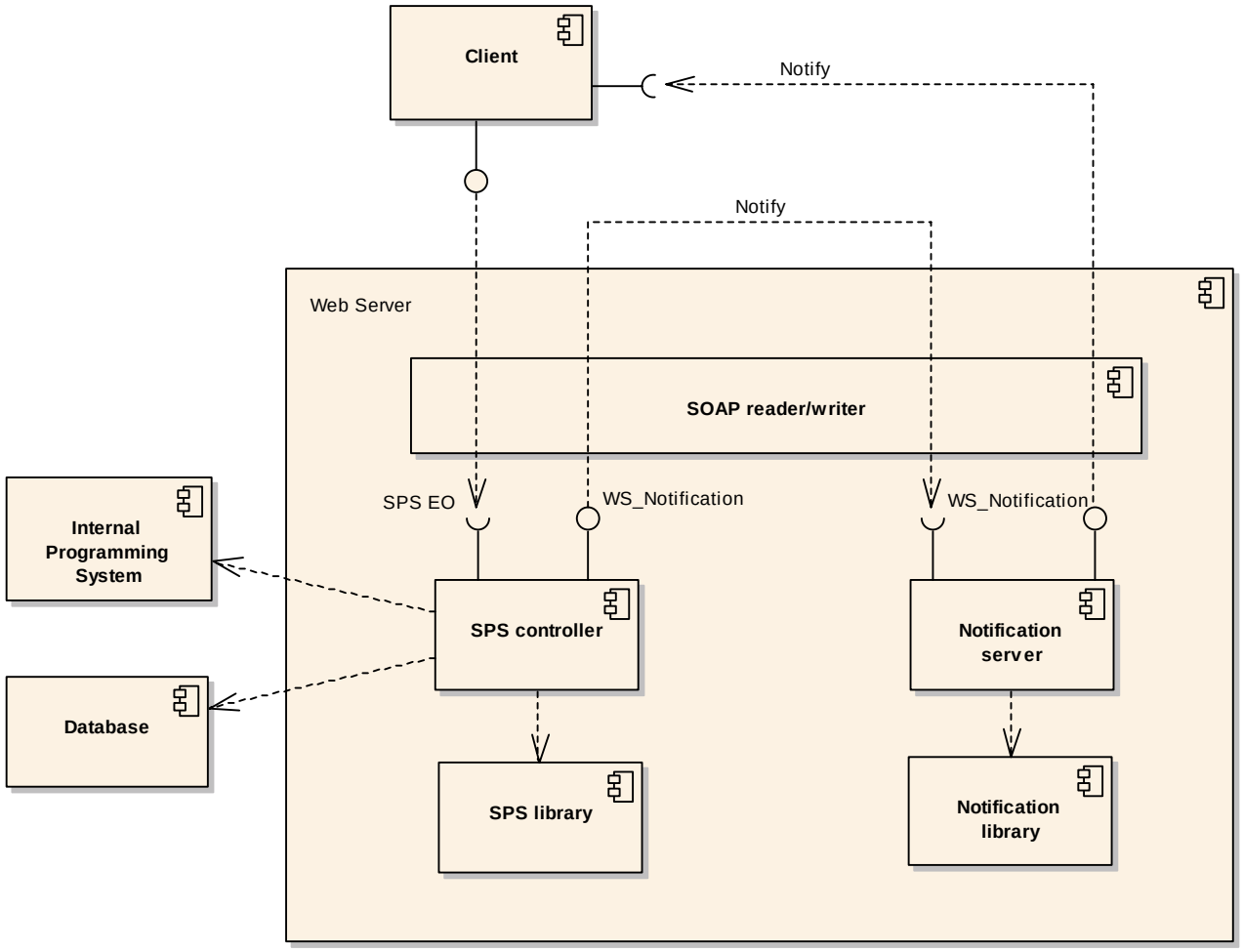

Figure 2: HMA-T SPS Interface Component Diagram

A more detailed description of each of the components identified above is presented in the table below.

Table 4: HMA-T SPS I/F Components

Component	Description
Web server	This component shall be an Apache Tomcat web server, which the clients can connect to with HTTP. The SOAP Reader/Writer and Library will be deployed within the Web Server.
SPS controller	Web Service that exposes an SPS EO interface. This is the core element that holds the logic of the system, in charge of the control and management of incoming requests and the creation of the appropriate responses.
SOAP Reader/Writer	This component is charged with the SOAP envelope extraction from incoming requests as well as appending the SOAP header to the built response prior to passing it to the web server for delivery to the client. Furthermore, along with the SPS Library it serializes and deserialises SOAP message contents.

Component	Description
SPS Library	The SPS library component is responsible for: ❑ Deserializing SPS requests coming from SPS clients into java objects that will be used by the SPS controller; ❑ Serializing java objects generated by SPS controller into SPS responses;
Notification Server	This component is responsible for sending notifications to the client when it has subscribed for some events. It is only needed in the SPOT SPS where processing of `GetFeasibility` or `Submit` messages is asynchronous
Ground Segment Internal Planning System	Planning system. It can be connected via a port or similar, and will receive and return information on the feasibility of a request. It is loosely coupled with the rest of the development.

4.3. High-Level System Context

The SPS Interface Server sits between a client requesting tasking of a sensor, and an existing planning system within a mission ground segment. The key component links, with a particular focus on the type of information that flows between them are shown below:

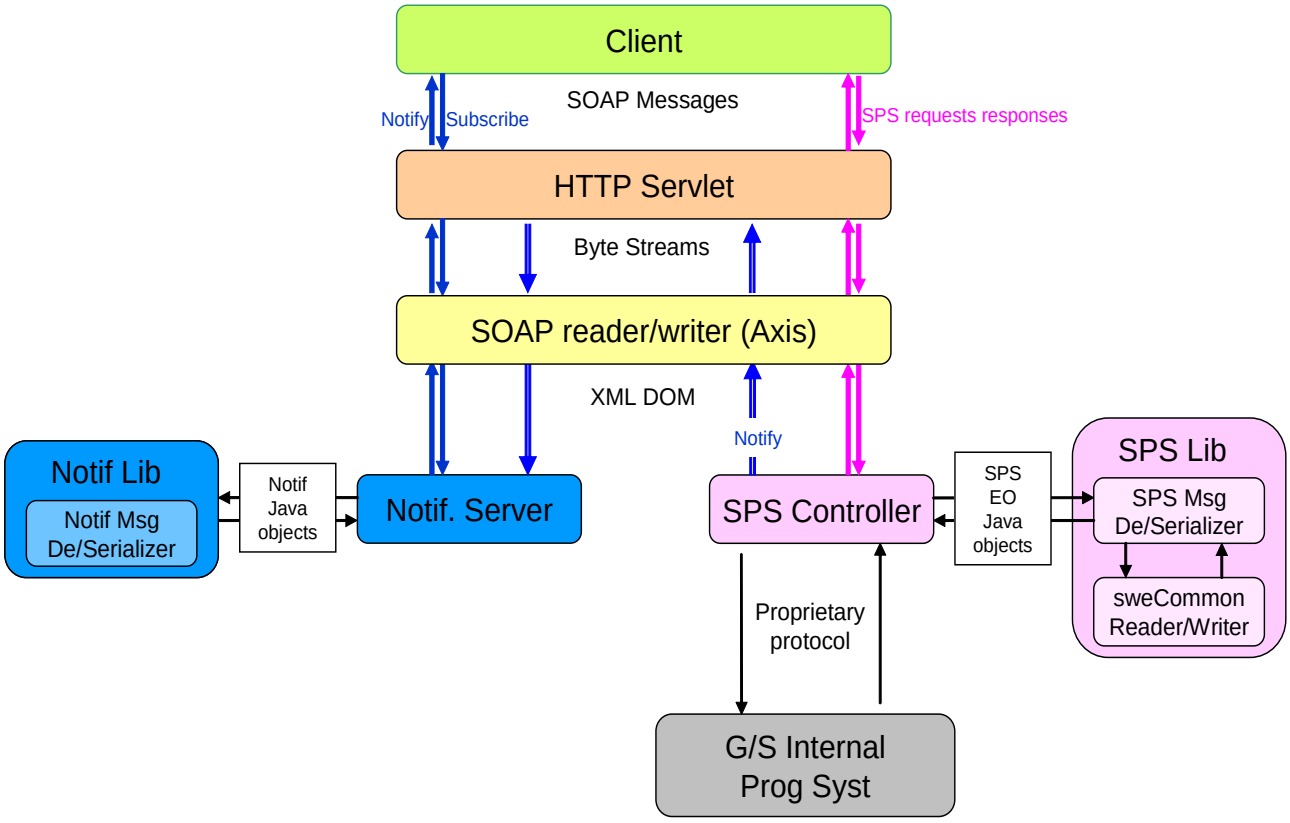

Figure 3: HMA-T SPS Interface Context diagram

4.4. High-Level Data Flow

The following diagram shows the data flow in case of the typical scenario for the SPOT SPS given in Section 4.1:

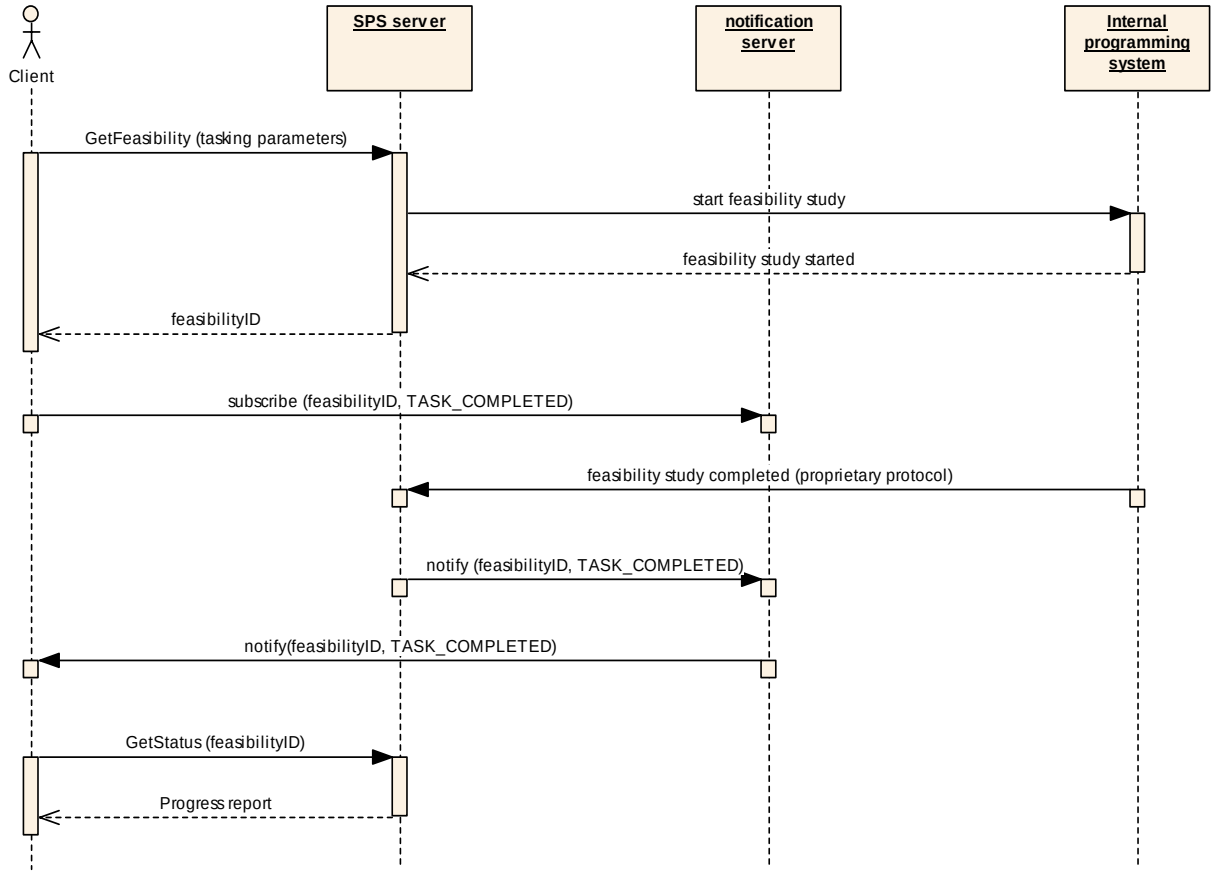

Figure 4: GetFeasibility asynchronous scenario (SPOT)

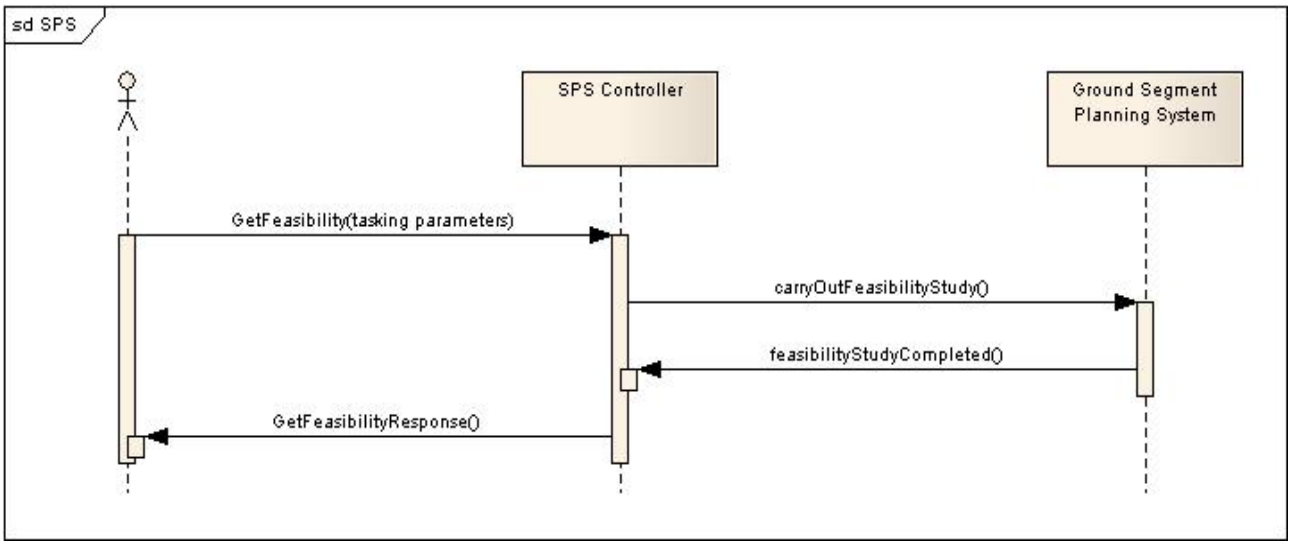

Figure 5: GetFeasibility synchronous scenario (DEIMOS)

4.5. Web Server Component Description

4.5.1. Type

The Web server is COTS software – Apache Tomcat.

4.5.2. Purpose

The Web server handles HTTP traffic, comprising the SOAP and HTTP KVP messages. These require different responses, and the web server is configured to pass them to the two relevant components.

4.5.3. Function

The Web server will accept connections from clients, and provide them with access to the SPS Interface. Many more details of Apache Tomcat can be found at http://tomcat.apache.org/.

4.5.4. Interfaces

- SOAP Reader/Writer

4.6. SPS Controller Component Description

4.6.1. Type

The SPS Controller is a Web Service implemented as a java servlet.

4.6.2. Purpose

The SPS Controller will manage the business logic for the processing of Web Service Client system requests, according to the SPS application profile - [OGC 07-018] version 1.0. It will receive these as Java Objects from the SOAP Reader/Writer component. In the case of the SPOT SPS there will also be Ground Segment planning system messages indicating that an asynchronous process has been completed and that notification is to be sent to the client.

4.6.3. Functions triggered by Web Service Clients

The following Use Case diagram shows that the SPS will process various message types, triggered by a client SOAP request as deserialized by the SOAP Reader/Writer.

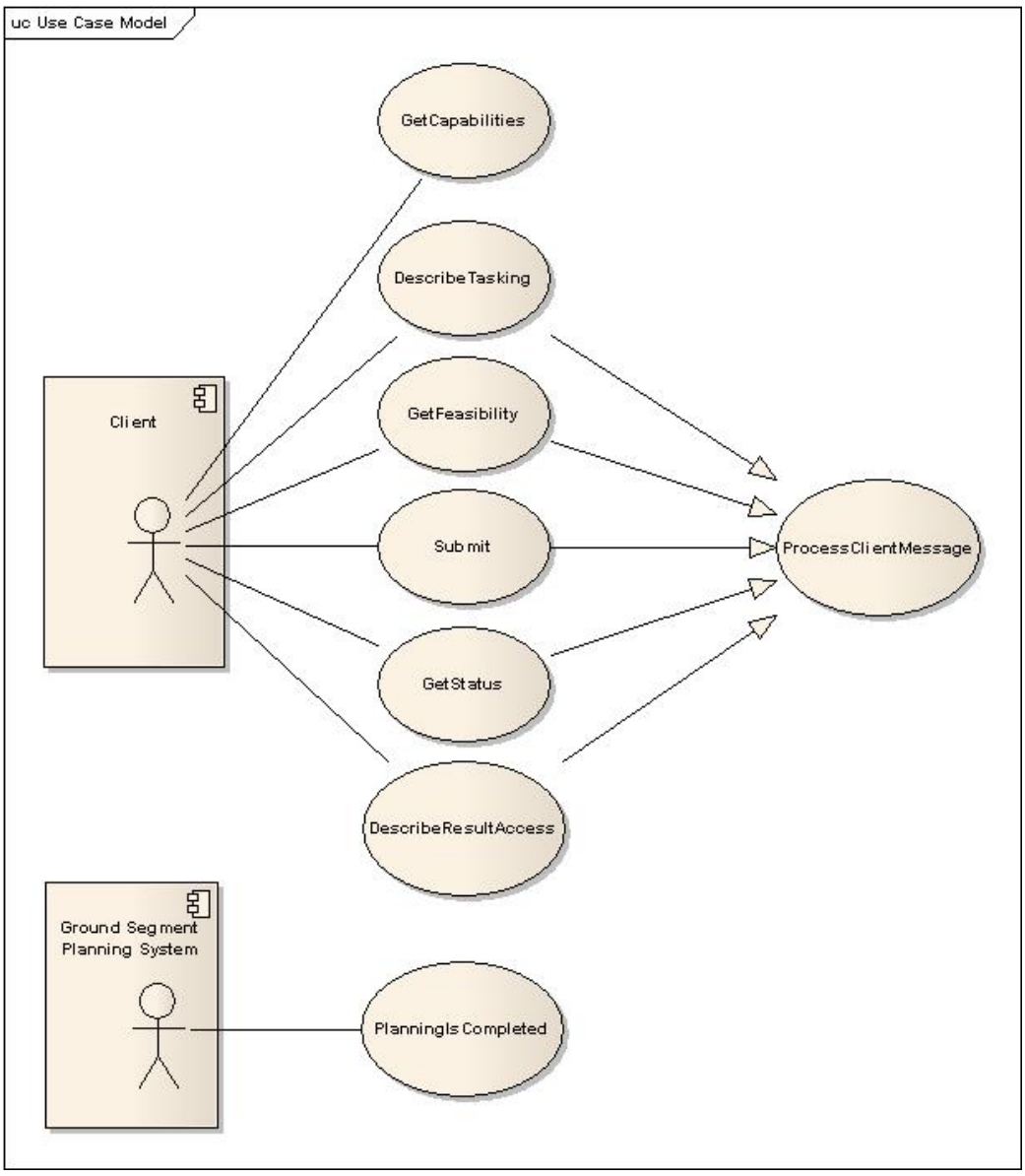

Figure 6: SPS Controller Use Cases

As can be seen, most of these specialize a single “Process Client message” Use Case, which consists of

1	Write to the database the relevant parts of the message (depends on message type, but should include which client sent it, and when, and the SOAP message unique identifier as a minimum)
2	Populate a Java object representing the response to be sent (synchronously) to the client. These range from simple acknowledgements or responses with the status of the operation (eg. *pending*) to XML structures describing scenes for acquisition. Full details of the set of messages, including XML Schema are found in draft 1.0 version of the [OGC 07-018] specification.
3	Send this “response” object to the SOAP Reader/Writer, which will send it as an XML message in the HTTP response.

There is a single extension point 2a, required for GetFeasibility and Submit requests where asynchronous "off-line" processing is needed by the Ground Segment. A synchronous response is always sent to the client, but the result may not be available synchronously. There is also a trigger to the internal planning system which must perform the task.

2a	Pass the Java object representing the request to the Ground Segment Planning System.

There is a separate process for the GetCapabilities Use Case:

1	Create an XML response to the request and send this as an XML message in the HTTP response. It is not a SOAP message and will not use the SOAP Reader/Writer

4.6.4. Functions triggered by Ground Segment Planning System

There is just one SPS controller function triggered by the internal Ground Segment sensor planning system. It is used in the SPOT case of asynchronous planning activity and will indicate that planning is completed and the client should be notified.

1	Write to the database the relevant information about the Tasking Feasibility. This will include everything required for responding to a future GetStatus message referencing this Feasibility request. Each feasibility / Submit request is uniquely identified by an ID, given by the server in the GetFeasibility / Submit response, that the client must refer to when sending a GetStatus request.
2	Trigger the Notification component to send a message to the relevant client indicating that the processing has completed. The information to be passed will be the client identifier, and the Feasibility Request identifier.

4.6.5. Data

The various parameters to store, are derived from the SPS Application profile, [OGC 07-018].

As well as the data used in every SPS there will be parameters specific to each sensor of each SPS. Even for a particular sensor, the SPS server is free to provide either a complete list of tasking parameters or just a subset.

4.6.6. Interfaces

- SOAP Reader/Writer: This will provide the input and output Java objects representing Client system requests and responses, according to the SPS application profile.
- Ground Segment Internal planning system: This will be given information for feasibility studies and will provide results.
- Database: This will be used to store various information.
- Notification Server (if needed): This will be triggered to send messages to clients when feasibility studies have completed.

4.7. Notification Server Component Description

4.7.1. Type

The Notification Component is a Web Service implemented as a java servlet.

4.7.2. Purpose

The Notification Component carries out the sending of messages to clients, to notify them about whichever events they have subscribed to, in the case of asynchronous planning system processing (SPOT SPS).

4.7.3. Function

Typically, the client sends a subscription request to the notification server for one or more events concerning a task. Then each time the notification server is notified by the SPS that an event related to the task occurred, it checks whether a client has subscribed to this event and if this is the case, sends a notify message to the client. The Notification Server will implement the WS-BaseNotification standard as defined in [RD Notifications].

In case of the client subscribes for a event that already occurred, the notification server automatically sends a Notify message containing that event to the client. Note that:

- the Notify message contains the date when the event occurred
- if the event occurred several times before the subscription was made, only one Notify is sent.

The available events that a client can subscribe to are listed in the SPS capabilities document.

4.7.4. Data

When receiving a notification coming from the SPS concerning a task, the Notification server must be able to retrieve the client that subscribed for the task:

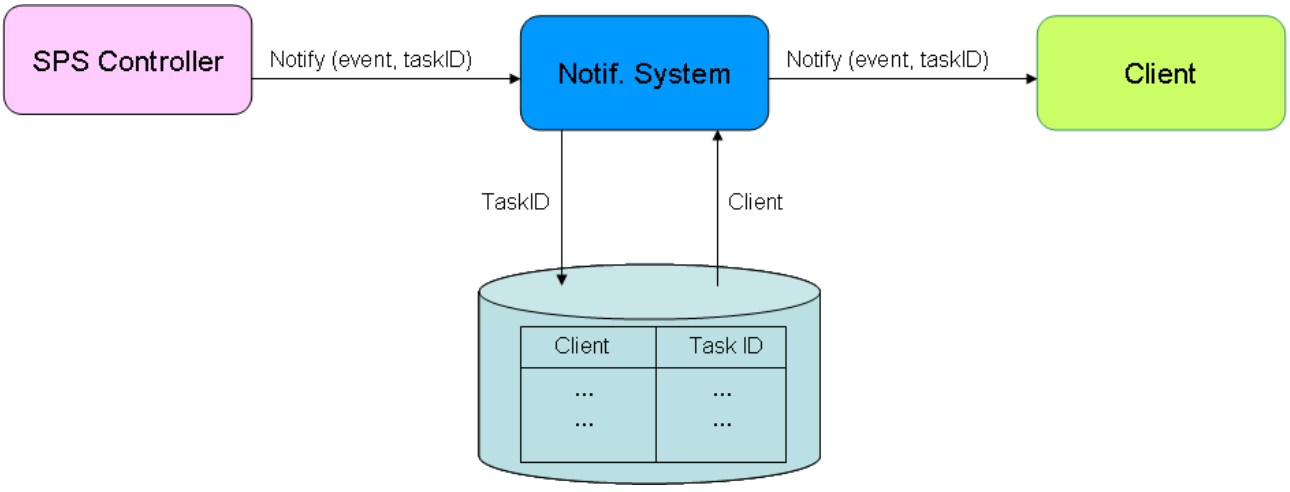

Figure 7: Notification Server

The Notification server stores taskID and clients' endpoint in a database.

4.7.5. Dependencies

The notification server is connected with the client and the SPS server. It uses the notification library provided by Spot Image in order to serialize and deserialize the exchanged messages.

4.7.6. Interfaces

- Coming from the client: subscription requests.
- Sending to the client: notifications.
- Coming from SPS controller: notifications.
- Notification Library: to assist in the (de)serialisation of the notification messages.

4.8. SOAP Reader/Writer Component Description

4.8.1. Type

The SOAP Reader/Writer consists of Java code automatically generated and run within the Axis 2 web service framework. A WSDL description and set of XSD Schemas will be written which together specify the web service, and then code generated by the Axis utility WSDL2Java.

4.8.2. Purpose

The component receives the SOAP messages from the web server and will provide SOAP messages back to it, following processing within the SPS Interface.

4.8.3. Function

This component deserializes the XML SOAP request messages into Java objects and provides them to the SPS Controller Component. It also provides a Java object to represent the appropriate response. When it receives this response object back from the SPS Controller it serializes the Java object into the equivalent SOAP message.

4.8.4. Interfaces

- Web Server: To provide and send out the SOAP messages.
- SPS Controller: To pass on and receive Java objects for the requests and responses.

4.9. SPS Library Component Description

4.9.1. Type

This component is a library of Java classes, which assist in deserialising SOAP messages into Java objects, and vice versa.

4.9.2. Purpose

The library is used by the SPS Controller when processing the messages that use sets of sweCommon parameters. These are the parameters used to make a specific request to the sensor, and are listed in detail in [OGC 07-018]. For the Radar sensor and Optical sensors we are simulating, those listed under the Radar Profile and Optical profile are particularly relevant.

4.9.3. Function

The Library performs validation as well as mapping from elements and attributes to Java objects. It supports the message processing.

4.9.4. Interfaces

- SPS Controller: To receive the whole set of parameters, and to return more specific Java objects representing them individually.

4.10. DEIMOS Ground Segment Internal Planning System

4.10.1. Type

The Ground Segment Planning System is a simulator written in Java, and using heavily the CFI library software. The CFI is written in C, but accessed via a JNI.

4.10.2. Purpose

The component simulates a simple planning system for one of the Radar instruments in the Sentinel missions.

4.10.3. Function

This component calculates what swathes of the Satellite Instrument's sensing will enable the requested coverage and return this information. It is representative of an automated internal Ground Segment planning system. The planning system component will respond synchronously and send the information to the SPS Controller, in order that it can be added to the database and the getFeasibilty response be sent to the client synchronously.

4.10.4. Interfaces

SPS Controller: This provides the feasibility request information, and the response must be sent to it

4.11. Spot Image Ground Segment Internal Planning System

4.11.1. Type

The Ground Segment Planning System is a simulator implemented as a library written in Java.

4.11.2. Purpose

The component simulates a planning system for one SPOT satellites and potentially other sensors.

4.11.3. Function

The component performs feasibility studies by taking into account:

- the climatology
- the orbitology
- the specificities of the SPOT sensors

When a calculation has finished then the planning system component will send the information to the SPS Controller, in order that it can be added to the database and notification be sent to the appropriate client.

4.11.4. Interfaces

- SPS Controller: This provides the feasibility request information, and the response must be sent to it.

5. TRACEABILITY MATRICES

5.1. Direct Traceability

Table 5: Requirement Traceability Matrix

System Requirement	System Components	Comments
SR-FUN-0010/1.0	Web Server, SPS Controller, SOAP Reader/Writer, SPS Library, Notification Server	
SR-FUN-0020/1.0	All.	This document only strictly covers the DEIMOS SPS I/F development, though the SPOT development uses a similar architecture
SR-FUN-0030/1.0	Web Server, SPS Controller, SOAP Reader/Writer, SPS Library, DEIMOS Ground Segment Internal Planning System	
SR-FUN-0040/1.0	Web Server, SPS Controller, SOAP Reader/Writer, SPS Library, SPOT Internal Planning Systems interface, Notification Server	
SR-FUN-0110/1.0	Web Server	
SR-FUN-0120/1.0	Web Server	
SR-FUN-0210/1.0	SPS Controller, SOAP Reader/Writer	
SR-FUN-0220/1.0	SPS Controller	
SR-FUN-0230/1.0	SPS Controller, Notification server	
SR-FUN-0310/1.0	Notification server	
SR-FUN-0320/1.0	Notification server, SPS Controller	
SR-FUN-0410/1.0	SOAP Reader/Writer	
SR-FUN-0420/1.0	SOAP Reader/Writer	
SR-FUN-0430/1.0	SOAP Reader/Writer, SPS Library	
SR-FUN-0510/1.0	SPS Library	
SR-FUN-0610/1.0	SPOT Internal Planning Systems interface	
SR-FUN-0710/1.0	DEIMOS Ground Segment Internal Planning System	
SR-FUN-0720/1.0	DEIMOS Ground Segment Internal Planning System	
SR-FUN-0730/1.0	DEIMOS Ground Segment Internal Planning System	
SR-PER-0010/1.0	All	
SR-PER-0020/1.0	Web Server, SOAP Reader/Writer, SPS Library, SPS Controller	

System Requirement	System Components	Comments
SR-INT-0010/1.0	Web Server	
SR-OPE-0010/1.0	All	Process, not software
SR-RES-0010/1.0	All	DEIMOS system
SR-RES-0020/1.0	All	SPOT system
SR-IMP-0010/1.0 -SR-IMP-0030/1.0	N/A	Process, not software
SR-CON-0010/1.0	N/A	Documentation, not software
SR-INS-0010/1.0	N/A	Documentation, not software. SPOT system will not to be installed at ESRIN.
SR-VVA-0010/1.0	N/A	Process, not software

5.2. Inverse Traceability

Table 6: Inverse Requirements Traceability Matrix

System Component	System Requirements	Comments
Web Server	SR-FUN-0010, SR-FUN-0020, SR-FUN-0030, SR-FUN-0040, SR-FUN-0110, SR-FUN-0120, SR-INT-0010, SR-PER-0010, SR-OPE-0010, SR-RES-0010, SR-RES-0020	
SPS Controller	SR-FUN-0010, SR-FUN-0020, SR-FUN-0030, SR-FUN-0040, SR-FUN-0210, SR-FUN-0220, SR-FUN-0230, SR-FUN-0320, SR-PER-0010, SR-OPE-0010, SR-RES-0010, SR-RES-0020	DEIMOS version does not need Notification Server interaction requirements
Notification server	SR-FUN-0010, SR-FUN-0020, SR-FUN-0040, SR-FUN-0230, SR-FUN-0310, SR-FUN-0320, SR-PER-0010, SR-OPE-0010, SR-RES-0020	
SOAP Reader/Writer	SR-FUN-0010, SR-FUN-0020, SR-FUN-0030, SR-FUN-0040, SR-FUN-0210, SR-FUN-0410, SR-FUN-0420, SR-FUN-0430, SR-PER-0010, SR-OPE-0010, SR-RES-0010, SR-RES-0020	
SPS Library	SR-FUN-0010, SR-FUN-0020, SR-FUN-0030, SR-FUN-0040, SR-FUN-0430, SR-PER-0010, SR-OPE-0010, SR-RES-0010, SR-RES-0020	
DEIMOS Ground Segment Internal Planning System	SR-FUN-0010, SR-FUN-0020, SR-FUN-0030, SR-FUN-0710, SR-FUN-0720, SR-FUN-0730, SR-PER-0010, SR-OPE-0010, SR-RES-0010	

System Component	System Requirements	Comments
SPOT Ground Segment Internal Planning interface	SR-FUN-0010, SR-FUN-0020, SR-FUN-0040, SR-FUN-0610, SR-PER-0010, SR-OPE-0010, SR-RES-0020	