HMA Follow On. Task 2: Feasibility Analysis Service

(Sensor Planning Service)

HMA-FO Feasibility Analysis Service

SYSTEM REQUIREMENTS DOCUMENT

Code : HMA-FO-DMS-TEC-SRD01-E-R
Issue : 1.0
Date : 26/01/10

	Name	Function	Signature
Prepared by	Reuben Wright Manuel Casado Corentin Guillo	DEIMOS Project Manager DEIMOS Project Engineer ASTRIUM Project Manager	
Reviewed by	Ángel Monge	DEIMOS Head of Section	
Approved by	Reuben Wright	DEIMOS Project Manager	
Signatures and approvals on original			

This page intentionally left blank

Document Information

Contract Data	
Contract Number:	22506/09/I-LG
Contract Issuer:	ESA

Internal Distribution		
Name	Unit	Copies
Reuben Wright	MOS	1
Ángel Monge	MOS	
Manuel Casado	MOS	1
Teresa García	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
Corentin Guillo	Astrium	1

Archiving	
Word Processor:	MS Word 2000
File Name:	HMA-FO-DMS-TEC-SRD01-10-E-R.doc

Document Status Log

Issue	Change description	Date	Approved
1.0	First version of the document	26/01/2010	

Table of Contents

List of Tables

List of Figures

1. INTRODUCTION

1.1. Purpose

This is the System Requirements Document (SRD) for the HMA-FO project **Task 2: Feasibility Analysis Service (Sensor Planning Service).**

The requirements cover the work corresponding to an open source Sensor Feasibility Reference Environment (SFRE) that will be used by ESA for the testing and demonstration of the SPS Profile for Earth Observation OGC 07-018 [RD 1]. There are requirements applicable to SF Client and Server systems which send and receive programming requests for EO products to support access to data from heterogeneous systems dealing with derived data products from satellite based measurements of the earth's surface and environment.

The document contains:

- Functional analysis of the interface, including approaches taken to solve specific problems identified during this analysis
- System requirements for the software components identified

1.2. Scope

This document is produced as part of the Technical Specification that shall be subject to review at PM1. Therefore, it is applicable to the project from PM1 onwards.

1.3. Document Structure

This document is organised as follows:

- Section 1 contains this introduction.
- Section 2 contains the list of applicable and reference documents, as well as the applicable standards to the project.
- Section 3 provides a general description of the SFRE.
- Section 4 describes the initial analysis made on the system prior to the specification of requirements.
- Section 5 presents the SFRE system requirements.
- Section 6 contains the traceability matrices between the system requirements and the requirements baseline.

1.4. Acronyms and Abbreviations

The acronyms and abbreviations used in this document are listed below:

Acronym	Meaning
AD	Architectural Design
AR	Acceptance Review
CCN	Contract Change Notice
CDR	Critical Design Review
CFI	Customer Furnished Item
CITE	Compliance & Interoperability Testing & Evaluation Initiative
CM	Configuration Management
COTS	Commercial Off-The-Shelf
CPU	Central Processing Unit
CR	Change Record
CTL	Compliance Testing Language
CVS	Concurrent Versions System
DB	Data Base
DDF	Design Definition File
DJF	Design Justification File
DMS	DEIMOS Space
ECSS	European Cooperation on Space Standardization
EO	Earth Observation
EO-DAIL	EO Data Access Integration Layer
EOEP	Earth Observation Envelope Programme
ESA	European Space Agency
ESTEC	European Space Technology Centre
FP	Final Presentation
FTP	File Transfer Protocol
GMES	Global Monitoring for Environment and Security
HMA	Heterogeneous Missions Accessibility
HMA-I	HMA Interoperability
HMA-T	HMA Testbed
HMI	Human Machine Interface
HW	Hardware
ICD	Interface Control Document
IDE	Integrated Development Environment
IPR	Intellectual Property Rights
IRD	Interface Requirements Document
ITT	Invitation to Tender
KOM	Kick-Off Meeting
N/A	Not Applicable

Acronym	Meaning
NCR	Non Conformance Report
OGC	Open Geospatial Consortium Inc.
OO	Object Oriented
PDR	Preliminary Design Review
QA	Quality Assurance
QR	Qualification Review
R&D	Research and Development
RB	Requirements Baseline
RID	Review Item Discrepancy
SAR	Synthetic Aperture Radar
SDE	Software Development Environment
SDP	Software Development Plan
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SoW	Statement of Work
SPOT	SPOT Image
SPR	Software Problem Report
SPS	Sensor Planning Service
SR	Software Requirement(s)
SRD	Software Requirements Document
SVV	Software Verification and Validation
SW	Software
TBC	To Be Confirmed
TBD	To Be Defined
TS	Technical Specification
UML	Unified Modelling Language
UR	User Requirement(s)
URD	User Requirements Document
WBS	Work Breakdown Structure
WP	Work Package
WPD	Work Package Description
XML	Extended Markup Language
XSD	XML Schema Definition
V&V	Verification & Validation
WBS	Work Breakdown Structure
WPD	Work Package Description

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
[AD 1]	SGSE-DFPR-EOPG-SW-08-0001	Statement of Work – HMA Follow on activities	1.2
[AD 2]	Appendix 3 to ESRIN/AO/1-5949/09/I-LG	Special Conditions Of Tender	-
[AD 3]	RES-POE/2009/34/LG/cb	Letter with the Invitation to Tender AO/1-5949/09/I-LG – HMA Follow On	-
[AD 4]	Appendix 2 to ESRIN/AO/1-5949/09/I-LG	Draft Contract – HMA Follow on activities	-
[AD 5]	ECSS-E-ST-40C	Software general requirements standard ECSS-E-ST-40C tailored to small software projects	-
[AD 6]	HMAFO-MOM-0001-SPB	Minutes of the negotiation meeting held at ESRIN on 24 June, 2009	-
[AD 7]	HMA-FO-DMS-PMD-PMP01-E-R	Project Management Plan	1.0
[AD 8]	HMA-FO-DMS-TEC-URD01-E-R	User Requirements Document	1.0
[AD 9]	ECSS-E-40 Part 1B	Software – Part 1: Principles and requirements	28/11/2003

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 1]	OGC 07-018	OpenGIS® Sensor Planning Service Application Profile for Earth Observation	2.0
[RD 2]	OGC 09-000	OpenGIS® Sensor Planning Service Implementation Standard	2.0
[RD 3]	OGC 06-126	Compliance Test Language (CTL)	Draft 0.1
[RD 4]	WS-BaseNotification	Web Services Base Notification OASIS Standard	1.3

[RD 5]	OGC 06-121r3	OpenGIS® Web Services Common Specification	1.1
[RD 6]	HMAT-TN-0001-IGN	HMA-T Phase 2 Testing Policy	1.1
[RD 7]	HMA-FO-DMS-LTR-001	DEIMOS' proposal for HMA-FO Feasibility Analysis Service	13/03/2009
[RD 8]	HMA-TN-ASU-EN-0001	Operational Scenarios Technical Note	1.8
[RD 9]	ISBN 0-201-57168-4	The Unified Modelling Language User Guide, Grady Booch, James Rumbaugh, Ivar Jacobson.	-

3. GENERAL DESCRIPTION

3.1. HMA Project Background and Perspectives

In 2005 the Agency launched the Heterogeneous Missions Accessibility (HMA) project. The aim of the HMA project was to involve the stakeholders, namely national space agencies, satellite or mission owners and operators, in a harmonization and standardization process of their ground segment services and related interfaces.

HMA Follow On Activities cover 4 tasks related to completing specifications and creating implementations.

Task 2: Feasibility Analysis Service (Sensor Planning Service) includes the design, development, testing and documentation of an open source Sensor Feasibility Reference Environment (SFRE) that shall be used by ESA for the testing and demonstration of the OpenGIS® Sensor Planning Service Application Profile for EO Sensors (OGC 07-018) [RD 1] specification. To support uptake, demonstration and testing of the profile an open source implementation of both sides (i.e. client and server) is to be developed.

3.2. OGC Specification Background and Perspectives

The functionality that OCG has targeted within a sensor web includes:

- Discovery of sensor systems, observations, and observation processes that meet our immediate needs
- Determination of a sensor's capabilities and quality of measurements
- Access to sensor parameters that automatically allow software to process and geolocate observations
- Retrieval of real-time or time-series observations and coverages in standard encodings
- Tasking of sensors to acquire observations of interest
- Subscription to and publishing of alerts to be issued by sensors or sensor services based upon certain criteria

Within the SWE group, the enablement of such sensor web service is pursued through the establishment of several standard interface definitions. The services are the following:

1. **Observations & Measurements (O&M)** – The general models and XML encodings for observations and measurements made using sensors.
2. **Sensor Model Language (SensorML)** – standard models and XML Schema for describing the processes within sensor and observation processing systems; provides information needed for discovery, georeferencing, and processing of observations, as well as tasking sensors and simulations.
3. **Sensor Observation Service (SOS)** – An open interface for a service by which a client can obtain observations and sensor and platform descriptions from one or more sensors.
4. **Sensor Planning Service (SPS)** – An open interface for a service by which a client can

- determine the feasibility of collecting data from one or more sensors or models
- submit collection requests to these sensors and configurable processes.

5. **Sensor Alert Service (SAS)** – An open interface for a web service for publishing of and subscribing to deliverable alerts from sensor or simulation systems.
6. **Web Notification Service (WNS)** – An open interface for a service by which a client may conduct asynchronous dialogues (message interchanges) with one or more other services.

The Sensor Planning Service Application Profile for Earth Observation in particular is concerned with:

1. Getting the list of parameters that can be specified for programming a specific sensor;
2. Verify the feasibility of the request that is going to be submitted;
3. Submit the request and then check its progress;
4. If necessary to cancel the submitted request;
5. Retrieve the sensor's acquired data.

This document, along with the updated OpenGIS® Sensor Planning Service Application Profile for Earth Observation [RD 1] specification, constitutes the Requirements Baseline for this Sensor Feasibility Reference Environment. The version of this profile specification we shall implement is 2.0.

3.3. SFRE Function and Purpose

The main purpose of the SFRE is the implementation and testing of the interfaces to Sensor Planning Services dedicated to the EO Sensor domain complying with specification OGC 07-018.

The implementation of two different systems shall be carried out:

- Sensor Feasibility Client (SF Client)
- Sensor Feasibility Server (SF Server)

They will both support the full range of operations described in the specification.

Users of the SF Client shall have the capability to define feasibility requests. These requests shall be addressed to the SF Server, which perform the corresponding feasibility analysis. The response(s) to a feasibility request shall include the number of images needed to satisfy the request with the according set of meshes and the necessary time frame (or exact time) for each image acquisition. All these feasibility analysis information shall be displayed to the user by the SF Client. The users shall be able to refine the previously submitted feasibility analysis request by limitation of the AOI, time frame, tasked sensors, requested missions or other tasking parameters. The users should also be able to select a number of meshes from a feasibility analysis response and submit a new set of feasibility requests (or other operations, such as Submit or Reserve) following this selection (in terms of AOI and time frame).

.

Note also that an SPS in general (and the SF Server in particular) shall be capable of computing alternatives for the tasking request checked for feasibility. These alternatives may slightly modify the tasking parameters contained in the request (e.g. to change the time frame of an intended task by a few minutes). These are to be presented to the users, but the SF Client shall be aware that because the SPS is a stateful service these alternatives might no longer be feasible when they send a tasking request with the parameters contained in on of the alternatives"]

The implementation of the SF Server interfaces shall be based on a common SPS EO library, representing a low-level module that deals with sweCommon operations.

The underlying sensor planning systems are simulated by use of the **Earth Explorer Mission Software CFI**. This CFI represents a specific COTS to the project and it is a collection of software functions performing accurate computations of mission related parameters for Earth Explorer missions. In particular, one set of functions is dedicated to computing time segments at which an Earth Explorer satellite, or one of its instruments is in view of various targets, including zones (defined as polygons or circles, on the earth ellipsoid or at a given altitude).

3.3.1.1. SPS Interface Implementation

Figure 3-1: SFRE interface use case diagram shows the UML context diagram including all operations of the interfaces. The entities “SF Client” and “SF Server” refer to any software component that can invoke/be invoked to carry out an SPS operation. In particular the systems built in this project will support the full set of operations.

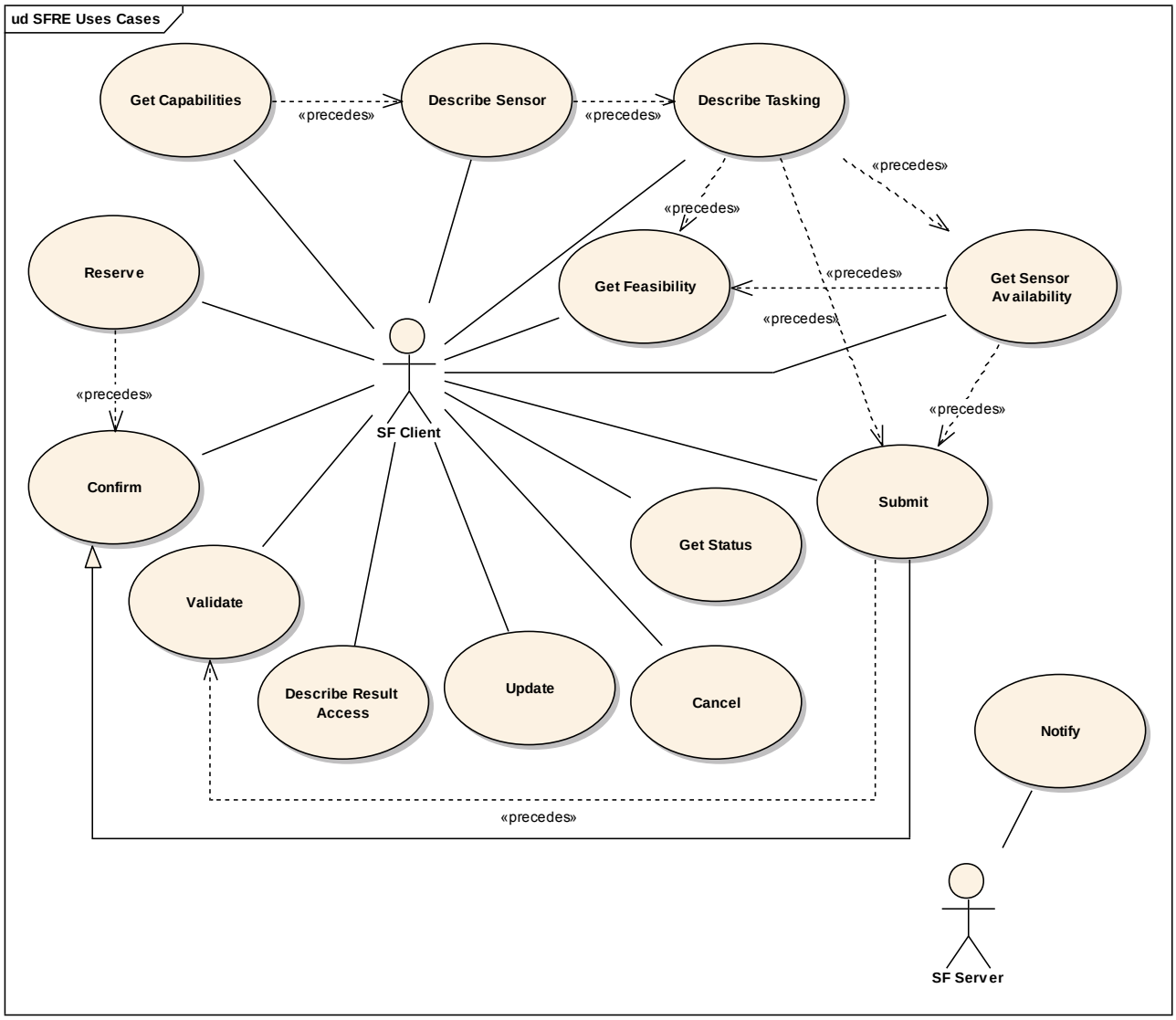

Figure 3-1: SFRE interface use case diagram

The Use Cases above correspond to the Operations specified in the SPS EO application profile which can be requested by a SF Client and performed by SF Server. In addition, the triggering of notifications based on certain internal state changes of the SF Server are indicated.

The SPS EO application profile specifies 13 operations that can be requested by a SF Client and performed by a SF Server. Those operations are:

a) GetCapabilities – This operation allows a client to request and receive service metadata (or Capabilities) documents that describe the abilities of the specific server implementation. This operation also supports negotiation of the specification version being used for client-server interactions. Moreover, the content section of this operation contains the list of sensorID provided by the service.

b) DescribeSensor – This operation allows the client to obtain a description of the sensors supported by the current SPS. The mission can decide on the amount of details provided in such a

description (The use of hyperlinks can help keep the initial document size small and simple while still allowing the client to go fetch more detailed information).

c) GetSensorAvailability (optional) – This operation provides information on the availability of the sensor.

d) Validate (optional) – Several acquisition attempts are sometimes necessary to obtain a satisfying result (case of optical satellites on zones with cloudy tendency for example). The Validate operation can be used by the customer to indicate to the server that an acquisition is satisfactory and thus to stop collecting new images for this area.

e) DescribeTasking – This operation allows a client to request the information that is needed in order to send GetFeasibility (for a feasibility study), Submit, Update and Reserve (for tasking the asset) requests. The response contains a description of the input (tasking parameters) and optionally the output parameters included in status reports.

f) GetFeasibility – This operation is to provide feedback to a client about the feasibility of a programming request. Depending on the sensor type offered by the SPS, the SPS server action may be as simple as checking that the request parameters are valid, and are consistent with certain business rules, or it may be a complex operation that calculates the usability of the sensor to perform a specific task at the defined location, time, orientation, calibration etc.

g) Submit – This operation submits the programming request. Dependent on the selected sensor, it may perform a simple modification of the sensor or start a complex mission.

h) GetStatus – This operation allows a client to receive information about the current status of the requested task. The response contains a progress report which content is defined by each service instance in the DescribeTasking response.

i) Cancel – This operation allows a client to request cancellation of a previously submitted task.

j) Update – This operation allows a client to update a previously submitted task.

k) DescribeResultAccess – This operation allows a client to retrieve information how and where data that was produced by the sensor can be accessed. The server response may contain links to any kind of data and not necessary through an OGC Web services nevertheless OGC Web services such as SOS, WMS, WFS or WCS are desirable.

l) Reserve – This operation reserves a task. A reservation lasts for a certain amount of time and can be committed during this timeframe.

m) Confirm – This operation is used to commit a reserved task. By committing a reserved task the SPS starts execution of the task.

The last two operations enable clients to reserve a task instead of directly submitting it. These operations have many similarities to other OGC® Web Services, including the WMS, WFS, and WCS. Many of these interface aspects that are common with other OWSs are thus specified in the OpenGIS® Web Services Common Implementation Specification [OGC 05-008] [RD 5].

The encoding of operation requests shall use HTTP GET with KVP encoding and HTTP POST with XML, SOAP, and/or KVP encoding as specified in [RD 5], [RD 2] and [RD 1]).

The SFRE will support all the operations, and at least one encoding method for each operation.

3.3.1.2. SPS Interface Testing

The complete verification and activities shall be performed upon the interfaces development.

- **Unit tests** shall be designed, implemented and executed to test each individual software component composing the system to be developed. Stubs shall be used in case of component interactions.
- **Integration tests** shall be designed, implemented and executed to mainly verify the interfaces between software components.
- Finally, **system tests** shall be designed, implemented and executed to validate the whole software.

It is to be noted, that normally system tests are also used for the acceptance, where a subset of tests is selected for the testing. System tests shall be carried out following the same approach as the unit and integration testing. Thus, test procedures shall be written in the same language as the SPS interface, and they shall test the involved operations by setting the input conditions, invoking the operation(s) subject to the test and verifying its response or outputs produced.

3.4. General Constraints

User management/authorisation/authentication is not considered in scope for these systems.

3.5. Environmental Considerations

For the **SF Server development** the following environment shall be used:

- Operating system: Linux.
- Development language: Java 6 using the JNI library for the interaction with the Earth Explorer CFI (written in C++), or a native Java implementation of Earth Explorer CFI.
- Essential libraries/software for the software development will be:
  - Apache Tomcat webserver
  - Axis2 SOAP framework
  - XML libraries for reading/writing XML files

For the **SF Client development** the following environment shall be used:

  - Operating System: Linux (Fedora 9)
- Essential libraries / software for the SF Client development are:
  - Apache Tomcat 6 webserver
  - Axis2 SOAP framework
  - XMLBeans for databinding
  - Development language: Java 6.
  - WorldWind Java library

- Google Web Toolkit

- For Configuration Management the following software shall be used:
  - CVS
  - SubVersion

4. FUNCTIONAL ANALYSIS

4.1. Introduction

This section presents the analysis made to create models that capture the essential requirements and characteristics of the SPS system. These models focus on *what* SPS components need to do, but leaves the details of *how* they will do it during the design phase.

4.2. Modelling Language

The Unified Modelling Language (UML) shall be used for describing the HMA-FO model. UML is a visual modelling language (not a method) for software systems, which considers two different aspects:

- **Static structure** – this describes what types of objects are important for modelling the system and how they are related.
- **Dynamic structure** – this describes the lifecycles of these objects and how they collaborate together to deliver the required system functionality.

The HMA-FO model shall be described using the following diagrams:

1. *Class diagrams* are used for describing the static view of the system. This document only identifies subsystems and the functionality allocated to them. Allocation of functions to classes is deferred to the design phase.
2. *Activity diagrams* are used for representing the dynamic behaviour of objects whose functionality is dominated by computations and data flows rather than states and events. Activity diagrams allow to model a process as a collection of activities and transition between those activities

The reader may find more detailed information on the Unified Modelling Language in [RD 9].

4.3. High-Level System Decomposition

4.3.1. HMA-FO class diagram

Figure 2: HMA-FO class diagram shows the initial class diagram for HMA-FO. There is a **Web Server**, name SPS Server to handle web communications, a **SOAP Reader/Writer** and **SPS library** for extracting and creating messages, and an **SPS controller** to manage the interactions with an **Internal SPS** and (if required) a **Notification Server**. The **Internal SPS** will interact with the **Earth Explorer Mission CFI** libraries through the **CFI Wrapper**.

The web communications are either handled directly over HTTP or as SOAP messages over HTTP. The two web communications components interface with any number of SPS Clients. The internal SPS system will simulate four missions: two of them will be optical missions and two of them radar mission.

The following diagram shows these components and their interactions.

The web server will be COTS, and the various components will be developed primarily in Java. Where there are defined C++ interfaces to an Internal SPS (Earth Explorer CFI based) there will be JNI translators to provide a Java interface.

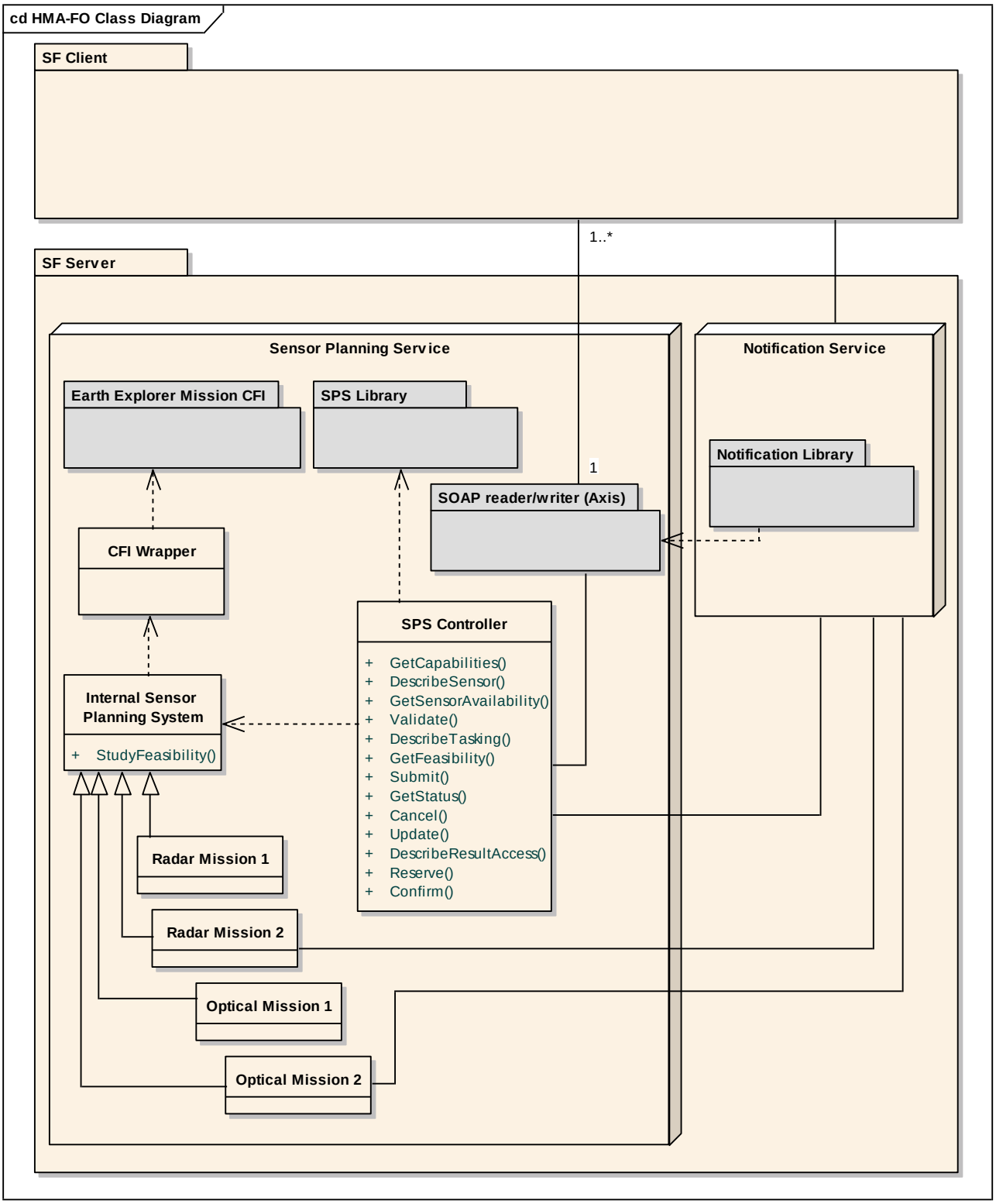

Figure 2: HMA-FO class diagram

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

Table 3: HMA-FO initial class decomposition

Component	Description
Sensor Planning Service	This component shall be an Apache Tomcat web server, where the clients can connect to, and where the various components conforming the SPS implementation will be deployed.
SPS Controller	This is the core element that holds the logic of the system, in charge of the control and management of incoming requests from their reception up to the delivery of the appropriate responses. It is a web application deployed within the web server.
Notification Service	This component, also deployed in an Apache Tomcat Web Server, is used to tell clients when the SPS has information for the client. The SPS controller can request the sending of a message to the subscribed clients. All the SPS operations are synchronous, in the sense of the server returning an immediate SOAP or XML response to the client. However there are certain operations where the internal processing may take some time. In those cases the synchronous response will return an identifier to use with an associated notification web service. The operator can then subscribe to get notifications (using the WS Base Notification standard) about the progress of the operation.
SOAP Reader/Writer	This component is an external library 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. It shall support versions 1.1 and 1.2of the SOAP specification.
SPS Library	The SPS library component is responsible for: ❑ Handling the extraction of data from SOAP message body elements, and write SOAP message bodies to pass back to the SOAP receiver component. ❑ Reading/writing sweCommon documents. ❑ Performing the deserialization/serialization of the SPS EO requests/responses. The data, once extracted, is available as Java classes, which are manipulated by the SPS Controller component to manage the interaction with the SPS Systems.
Internal Sensor Planning System	This component will perform feasibility studies based on Earth Explorer CFI technology. It will be configured in order to expose four simulated missions: ❑ **Radar Mission 1:** Radar Synchronous Mission. ❑ **Radar Mission 2:** Radar Asynchronous Mission. It makes use of

Component	Description
	the Notification Service to provide an asynchronous response to the client. ❑ **Optical Mission 1:** Optical Synchronous Mission. ❑ **Optical Mission 2:** Optical Asynchronous Mission. It makes use of the Notification Service to provide an asynchronous response to the client. The simulation will be done via a number of configuration files that will be parsed by the Internal Planning System and it will include the modeling of a number of conflicts/constraints as listed here (and described with more detail in section 5.8): ❑ Sensor unavailability ❑ Weather conditions ❑ Station unavailability These unavailability conditions shall be configurable by the edition of the associated configuration file. *Figure 3: Simulation of multiple missions* shows the simulation of multiple missions based on Earth Explorer CFI configurations and the modelling of different conflicts via Environment Configuration files.
CFI Wrapper	Based on JNI (framework that allows Java code running on a Java Virtual Machine to call native applications written in other languages) this component encapsulates the interactions with the Earth Explorer CFI library, written in C++.
Earth Explorer Mission CFI	The Earth Explorer Mission CFI software is a collection of classes and methods performing accurate computations of mission related parameters for Earth Explorer missions.
SF Client	The SF Client is based on Java technologies. It will allow the users, through a rich interface based on Google Web Tool Kit and World Wind Java, to define feasibility analysis requests, send them to the SF Server and to display the according responses for order submission.

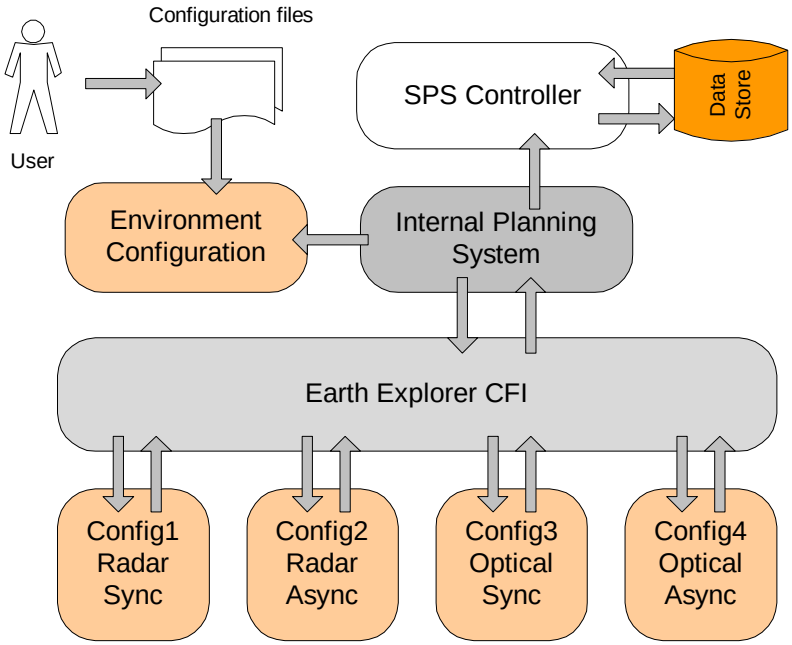

Figure 3: Simulation of multiple missions

4.4. Sequence Diagram

The following diagrams show the messages exchanged by the architectural components in order to attend http/Get with KVP and http/Post with SOAP requests.

The Apache web server shall keep listening to requests coming from clients and shall route them to the SPS Controller component, which shall be responsible to attend to them. However, the first thing to deal with is the extraction of the SOAP header and serialise the payload contents, so that the data is managed through Java classes. These activities are carried out by the SOAP Reader Writer and SPS Library respectively. Afterwards, the SPS Controller shall interact with the internal SPS exchanging information to satisfy the original request. As soon as this is done, the data is processed and translated into the XML messages as defined in [RD 1], and finally, the appropriate identifiers and endpoint are appended to send the message back. As for the message reception, the serialization of Java classes into XML message is handled by the SPS Library and the second stage (adding header and passing to web server) by the SOAP Reader/Writer.

The examples below show:

a representative case for the basic HTTP GET exchange, where the XML response is sent immediately back to the client via the same HTTP connection.

a representative case for synchronous SOAP message exchange, where the response is sent immediately back to the client via the same HTTP connection.

a representative case for asynchronous message processing, where a response is sent immediately back to the client via the same HTTP connection but further activity generates a Notify message. This then triggers the client to send a getStatus to receive the outcome of the processing.

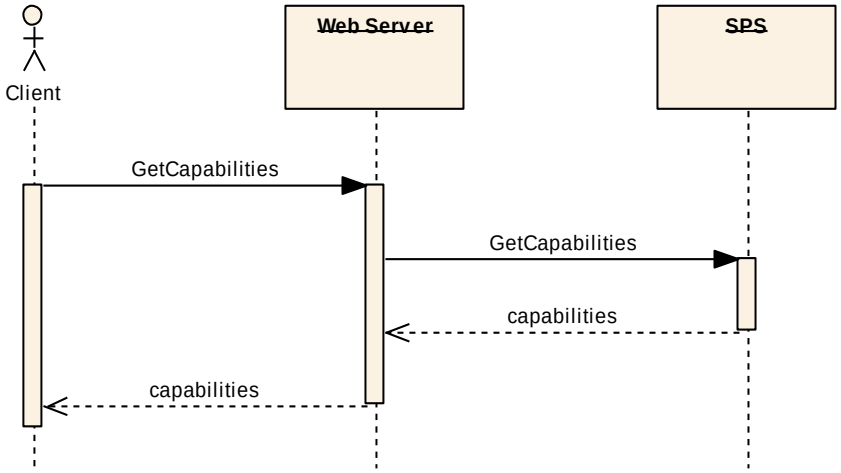

Figure 4: Sequence diagram for the HTTP GET GetCapabilities operation

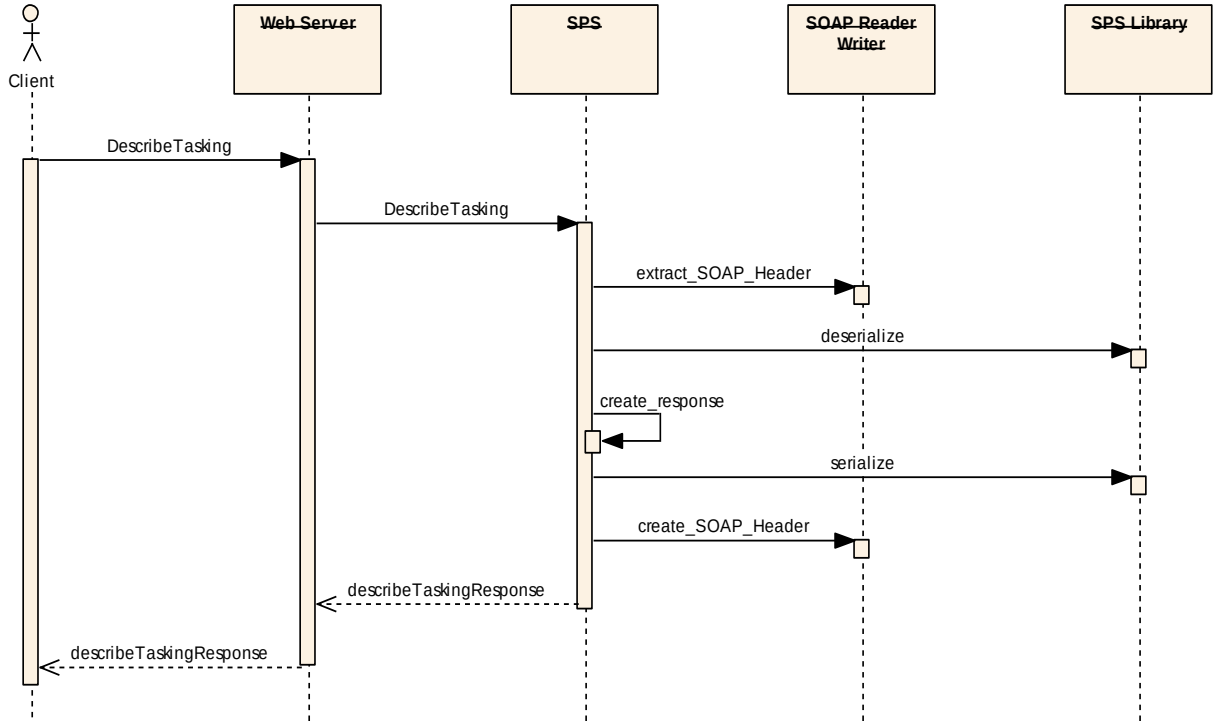

Figure 5: Sequence diagram for a HTTP POST operation

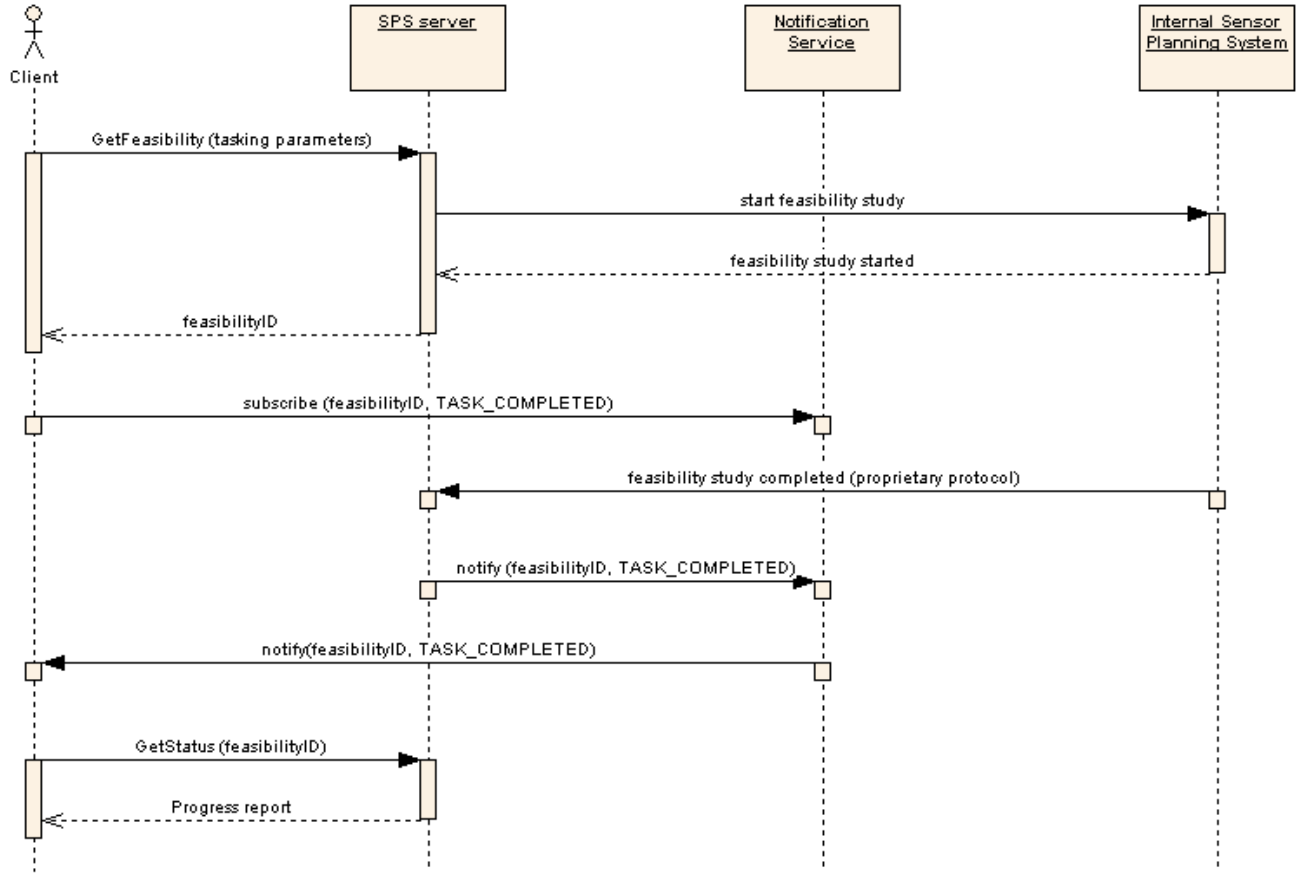

Figure 6: Use of Notification Service in asynchronous processing scenarios

5. SYSTEM REQUIREMENTS SPECIFICATION

5.1. Requirement Naming Conventions

The following conventions will be used for the derived system requirement naming:

All system requirements will be named as

SR–XXX–NNNN/VER

where:

XXX represents the type of software requirement, which for HMA-T, are the following:

- **CON** – Software Configuration and Delivery Requirements
- **FUN** – Functional
- **IMP** – Design and Implementation Constraints
- **INS** – Installation
- **INT** – Interfaces
- **OPE** – Operational
- **PER** – Performance
- **RES** – Resources (Hardware and Software)
- **VVA** – Verification, Validation and Acceptance

NNNN Is a number providing an ordering within each requirement type. It starts at 0010 and two consecutive requirements are increased in 10 to allow the introduction of additional requirements in later versions of the document.

VER Is the issue of this document where the requirement was introduced or last changed.

Each requirement is presented in a tabular form constituted by four fields:

3. System requirement identifier
4. Validation method for the requirement:
   - **Test** (T) – Execution of the element under certain conditions to check the outputs corresponding to particular inputs;
   - **Inspection** (I) – Exhaustive evaluation of the code by manual reading;
   - **Analysis** (A) – Deduction method applied to documentation, code, test results, etc; it is partially or totally automated;
   - **Review** (R) – Review of project documentation.
5. Requirement text

5.2. Functional Requirements

5.2.1. General

SR-FUN-0010/1.0 T

The Sensor Feasibility Client and Server shall provide compliant implementation of the **GetCapabilities** operation following the OpenGIS® Sensor Planning Service Application Profile for EO Sensors (OGC 07-018) [RD 1] specification.

This operation allows a client to request and receive service metadata (or Capabilities) documents that describe the abilities of the specific server implementation. This operation also supports negotiation of the specification version being used for client-server interactions. Moreover, the content section of this operation contains the list of sensorIDs provided by the service

SR-FUN-0020/1.0 T

The Sensor Feasibility Client and Server shall provide compliant implementation of the **DescribeSensor** operation following the OpenGIS® Sensor Planning Service Application Profile for EO Sensors (OGC 07-018) [RD 1] specification.

This operation allows the client to obtain a description of the sensors supported by the current SPS. The mission can decide on the amount of details provided in such a description (The use of hyperlinks can help keep the initial document size small and simple while still allowing the client to go fetch more detailed information).

SR-FUN-0030/1.0 T

The Sensor Feasibility Client and Server shall provide compliant implementation of the **GetSensorAvailability** operation following the OpenGIS® Sensor Planning Service Application Profile for EO Sensors (OGC 07-018) [RD 1] specification.
This operation provides information on the availability of the sensor.

SR-FUN-0040/1.0 T

The Sensor Feasibility Client and Server shall provide compliant implementation of the **Validate** operation following the OpenGIS® Sensor Planning Service Application Profile for EO Sensors (OGC 07-018) [RD 1] specification.
Several acquisition attempts are sometimes necessary to obtain a satisfying result (case of optical satellites on zones with cloudy tendency for example). The Validate operation can be used by the customer to indicate to the server that an acquisition is satisfactory and thus to stop collecting new images for this area.

SR-FUN-0050/1.0 T

The Sensor Feasibility Client and Server shall provide compliant implementation of the **DescribeTasking**

operation following the OpenGIS® Sensor Planning Service Application Profile for EO Sensors (OGC 07-018) [RD 1] specification.
This operation allows a client to request the information that is needed in order to send GetFeasibility (for a feasibility study), Submit, Update and Reserve (for tasking the asset) requests. The response contains a description of the input (tasking parameters) and optionally the output parameters included in status reports.

SR-FUN-0060/1.0 T

The Sensor Feasibility Client and Server shall provide compliant implementation of the **GetFeasibility** operation following the OpenGIS® Sensor Planning Service Application Profile for EO Sensors (OGC 07-018) [RD 1] specification.
This operation is to provide feedback to a client about the feasibility of a programming request. Depending on the sensor type offered by the SPS, the SPS server action may be as simple as checking that the request parameters are valid, and are consistent with certain business rules, or it may be a complex operation that calculates the usability of the sensor to perform a specific task at the defined location, time, orientation, calibration etc.

SR-FUN-0070/1.0 T

The Sensor Feasibility Client and Server shall provide compliant implementation of the **Submit** operation following the OpenGIS® Sensor Planning Service Application Profile for EO Sensors (OGC 07-018) [RD 1] specification.
This operation submits the programming request. Dependent on the selected sensor, it may perform a simple modification of the sensor or start a complex mission.

SR-FUN-0080/1.0 T

The Sensor Feasibility Client and Server shall provide compliant implementation of the **GetStatus** operation following the OpenGIS® Sensor Planning Service Application Profile for EO Sensors (OGC 07-018) [RD 1] specification.
This operation allows a client to receive information about the current status of the requested task. The response contains a progress report which content is defined by each service instance in the DescribeTasking response.
Depending on the state of the task the GetStatus operation will return a StatusReport, FeasibilityReport or ReservationReport as defined in section 7.4.6.4 of OpenGIS® Sensor Planning Service Implementation Standard [RD 2].

SR-FUN-0090/1.0 T

The Sensor Feasibility Client and Server shall provide compliant implementation of the **Cancel** operation following the OpenGIS® Sensor Planning Service Application Profile for EO Sensors (OGC 07-018) [RD 1] specification.
This operation allows a client to request cancellation of a previously submitted task.

SR-FUN-0100/1.0 T

The Sensor Feasibility Client and Server shall provide compliant implementation of the **Update** operation following the OpenGIS® Sensor Planning Service Application Profile for EO Sensors (OGC 07-018) [RD 1] specification.

This operation allows a client to update a previously submitted task.

SR-FUN-0110/1.0 T

The Sensor Feasibility Client and Server shall provide compliant implementation of the **DescribeResultAccess** operation following the OpenGIS® Sensor Planning Service Application Profile for EO Sensors (OGC 07-018) [RD 1] specification.
This operation allows a client to retrieve information how and where data that was produced by the sensor can be accessed. The server response may contain links to any kind of data and not necessary through an OGC Web services nevertheless OGC Web services such as SOS, WMS, WFS or WCS are desirable.

SR-FUN-0120/1.0 T

The Sensor Feasibility Client and Server shall provide compliant implementation of the **Reserve** operation following the OpenGIS® Sensor Planning Service Application Profile for EO Sensors (OGC 07-018) [RD 1] specification.
This operation reserves a task. A reservation lasts for a certain amount of time and can be committed during this timeframe.

SR-FUN-0130/1.0 T

The Sensor Feasibility Client and Server shall provide compliant implementation of the **Confirm** operation following the OpenGIS® Sensor Planning Service Application Profile for EO Sensors (OGC 07-018) [RD 1] specification.
This operation is used to commit a reserved task. By committing a reserved task the SF Server simulates starting execution of the task.

SR-FUN-0140/1.0 T

The Sensor Feasibility Client and Server shall support both Coverage Order and Acquisition Order as it is detailed in section 8.1.1. of the OpenGIS® Sensor Planning Service Application Profile for EO Sensors (OGC 07-018) [RD 1] specification.

SR-FUN-0150/1.0

In case the SF Server encounters an error while performing one of its operations, it shall return an exception report message as specified in section 7.3 of OpenGIS® Sensor Planning Service Implementation Standard [RD 2].

SR-FUN-0160/1.0 T

There will be four simulated systems exposed by the server. They will have different sensor identifiers and so will be capable of being tasked independently. The simulations are all using the Earth Explorer CFI library and will essentially consist of four different configurations (mission type, swath type, orbit information, etc.) for the library.

SR-FUN-0170/1.0 T

The SF Server shall be configured to simulate the presence of four missions:

- Optical asynchronous mission whichmakes use of a Notification Service
- Optical synchronous mission
- Radar asynchronous mission which makes use of a Notification Service
- Radar synchronous mission

SR-FUN-0180/1.0 T

The SF Server shall maintain a list of states a request/order can be in stored in the *database* as defined in section 7.4.1.6. of OpenGIS® Sensor Planning Service Implementation Standard [RD 2].

SR-FUN-0190/1.0 T

The SF Server shall manage state transitions as specified in Figure 13 (section 7.4.1.6.) of OpenGIS® Sensor Planning Service Implementation Standard [RD 2]. Any change of state will be saved and needed notifications will be triggered.

SR-FUN-0200/1.0 T

The SF Client shall be configured to interface the four fixed missions exposed by the SF Server

SR-FUN-0210/1.0 T

The SF Client shall allow the users to task one or more integrated missions in the same feasibility analysis request

SR-FUN-0220/1.0 T

The SF Client shall allow the users to define the requests parameters according to the interfaced missions

SR-FUN-0230/1.0 T

The SF Client shall allow the users to display the response(s) from a feasibility analysis request performed by the SF Server

SR-FUN-0240/1.0 T

The SF Client shall allow the users to display the requests parameters in addition to the feasibility analysis responses

SR-FUN-0250/1.0 T

The SF Client shall allow the users to refine the previously submitted feasibility analysis request accordingly to the related received feasibility analysis response by:

- Selecting one or more meshes

And / Or

- Constraining key parameters (like ROI or period)

Before re-submitting a feasibility analysis study.

5.2.2. Web server

SR-FUN-0310/1.0 R

SPS shall use the Apache/Tomcat Web server for hosting the SPS implementation.

SR-FUN-0320/1.0 I

The Web server shall be accessible for protocols as needed for the required operations – HTTP GET with information in Keyword-Value pairs and HTTP POST with information in SOAP messages. It shall support versions 1.1 and 1.2 of the SOAP specification.

5.2.3. SPS Controller

SR-FUN-0410/1.0 T

SPS Controller shall process the received messages in line with the (OGC 07-018) [RD 1] specification. This includes synchronous responses for each message type.

SR-FUN-0420/1.0 I

SPS Controller shall populate the Java classes representing the SOAP responses, and pass these back to the SOAP Reader/Writer to be sent to the client synchronously.

SR-FUN-0430/1.0 T

If necessary for any change of state of a request or task the SPS Controller shall send information to the Notification server by sending *Notify* SOAP messages to alert a client that a task request has been processed successfully.

5.2.4. Notification Service

SR-FUN-0510/1.0 T

The Notification Service shall provide a mechanism for users to subscribe to notifications when certain

events occur during the lifecycle of a task.

SR-FUN-0520/1.0 T

The notification process shall be done via WS-Notification protocol [RD 4]

SR-FUN-0530/1.0 T

The Notification Service shall provide a predefined list of topics that correspond with all possible task status transitions. The following are identified in OpenGIS® Sensor Planning Service Implementation Standard [RD 2]:

- **TASK_FEASIBLE***: Events in this topic are generated when a task feasibility request goes from the *PENDING* state to the *FEASIBLE* state
- **TASK_ACCEPTED***: Events in this topic are generated when a task goes from the *PENDING* state to the *ACCEPTED* state
- **TASK_REJECTED***: Events in this topic are generated when a task goes from the *PENDING* state to the *REJECTED* state
- **TASK_COMPLETED***: Events in this topic are generated when a task goes from the *ACCEPTED* state to the *COMPLETED* state
- **TASK_FAILED***: Events in this topic are generated when a task goes from the *ACCEPTED* state to the *FAILED* state
- **TASK_RESERVED**: Events in this topic are generated when a task goes from the *PENDING* state to the *RESERVED* state
- **TASK_EXPIRED**: Events in this topic are generated when a task goes from the *RESERVED* state to the *EXPIRED* state
- **TASK_CANCELED***: Events in this topic are generated when a task goes from the *ACCEPTED* or *RESERVED* state to the *CANCELLED* state (cancelled either by server or client)

These topics do not correspond to state transitions, but events can be generated for these topics when a task is in the ACCEPTED state. These are described in the OpenGIS® Sensor Planning Service Application Profile for EO Sensors (OGC 07-018) [RD 1] specification:

- eo:SEGMENT_SCHEDULED: Events in this topic are generated when a new segment has been scheduled for a given task

- eo:SEGMENT_ACQUIRED*: Events in this topic are generated when a new segment has been acquired for a given task

- eo:SEGMENT_VALIDATED: Events in this topic are generated when a segment acquired for a given task has been validated either by the user or by the satellite provider

- eo:TASK_IN_PROGRESS*: Events in this topic are generated when a task goes from the ACCEPTED state to the IN_PROGRESS state (see below)

All topics marked with an asterisk shall be advertised by the Notification Service, and corresponding notification messages shall be generated and sent to subscribers.

5.2.5. SOAP Reader/Writer

SR-FUN-0610/1.0 I

SPS SOAP Reader/Writer shall extract the SOAP headers from the incoming SPS requests. It shall support version 1.1 of the SOAP specification.

SR-FUN-0620/1.0 I

SPS SOAP Reader/Writer shall append a SOAP header to the serialized information constituting a response to a given SPS request. It shall support version 1.1 of the SOAP specification.

5.2.6. SPS Library

SR-FUN-0710/1.0 I

The SF Server shall use the SPS Library for deserialization of the incoming requests and the serialisation of the generated responses.

SR-FUN-0720/1.0 I

SPS Library shall extract the SOAP body and shall pass it to the SPS Controller to trigger further processing. This extraction shall involve the creation and population of Java classes representing the same structures as appear in the SOAP input and output messages.

5.2.7. Internal Sensor Planning System

SR-FUN-0810/1.0 I

The Internal Sensor Planning System shall perform feasibility studies by using the Earth Explorer CFI as a COTS Satellite Programming System for the implementation.

SR-FUN-0820/1.0 I

The implementation of the interface related to the Earth Explorer CFI shall use either the JNI library as a bridge between the C++ and Java languages, or a native Java implementation of Earth Explorer CFI.

SR-FUN-0840/1.0 T

The Internal Planning System shall consider environmental configuration for unavailability scenarios when performing feasibilities studies. The configuration of these conflicts/constraints shall be done via a set of user-editable configuration files.

5.2.8. SF Client

SR-FUN-0910/1.0 T

The SF Client should be based on WorldWind Java and Google Web Toolkit technologies

SR-FUN-0920/1.0 T

The SF Client shall be executable in a common web browser (Mozilla Firefox, IE) run on a Windows (WP, Vista or Seven) based operating system

SR-FUN-0930/1.0 T

The SF Client shall display a 3D representation of the earth as the cartographic component of the application

SR-FUN-0940/1.0 T

The SF Client shall be interfaced to the SF Server following the OpenGIS® Sensor Planning Service Application Profile for EO Sensors (OGC 07-018) [RD 1] specification

SR-FUN-0950/1.0 T

The SF Client shall allow the users to define a feasibility analysis request based on the AOI, time frame and tasked mission parameters

SR-FUN-0960/1.0 T

The SF Client shall allow the users to define or not technical parameters according to the tasked missions

SR-FUN-0970/1.0 T

The SF Client shall allow the users to task one or more mission in the same feasibility analysis request

SR-FUN-0980/1.0 T

The SF Client shall allow the users to display the feasibility analysis response from the SF Server

SR-FUN-0990/1.0 T

The SF Client shall allow the users to display the feasibility request parameters related to the feasibility analysis response received

SR-FUN-1000/1.0 T

The SF Client shall allow the users to display one or more meshes from the feasibility analysis response by selecting the related image acquisition

SR-FUN-1010/1.0 T

The SF Client shall allow the users to request the status of the feasibility analysis request

SR-FUN-1030/1.0 T

The SF Client shall allow the users to refine the feasibility analysis request (from the previous feasibility analysis response) by constraining the key parameters (ROI or period) or selecting one or more meshes for submitting a new feasibility analysis.

SR-FUN-1050/1.0 T

The SF Client shall manage the Acquisition Order and the Coverage Order transparently for the users

5.3. Performance Requirements

SR-PER-0010/1.0 T

SFRE shall be capable of being deployed on a single server.

SR-PER-0020/1.0 T

SF Server shall respond rapidly enough to all message receptions that sensible time-out parameters can be set for any client software. 60 seconds is the anticipated value, although this may be changed during testing if necessary.
This is the benchmark when a single message is being processed at once. In case of multiple clients connecting at the same moment it is permissible (though not expected) that one should be given a time out by the SF Server in the understanding that the client will retry later.

5.4. Interface Requirements

SR-INT-0010/1.0 T

The SFRE system (SF Server and SF Client) shall be deployed on servers within DEIMOS/ASTRIUM facilities or at ESRIN.

SR-INT-0020/1.0 T

The encoding of the operation requests shall use HTTP GET with KVP encoding and HTTP POST with XML, SOAP (1.1 and 1.2), and/or KVP as specified in Clause 11 of [RD 5]. Table 4: Operations request encoding summarizes the SPS EO Service operations and their encoding methods as defined in [RD 1].

Table 4: Operations request encoding

Operation Name	Request Encoding
GetCapabilities (mandatory)	KVP and optional SOAP/XML
DescribeSensor	SOAP/XML and optional KVP
DescribeTasking (mandatory)	SOAP/XML and optional KVP
GetFeasibility	SOAP/XML
Submit (mandatory)	SOAP/XML
Update	SOAP/XML
GetStatus (mandatory)	SOAP/XML and optional KVP
Cancel	SOAP/XML and optional KVP
DescribeResultAccess (mandatory)	SOAP/XML and optional KVP

Update	SOAP/XML and optional KVP
GetSensorAvailability	SOAP/XML and optional KVP
Reserve	SOAP/XML
Confirm	SOAP/XML and optional KVP

5.5. Operational Requirements

SR-OPE-0010/1.0 R

SFRE is not intended to be an operational system. The developed systems shall be maintained only on a best-effort basis.

5.6. Resources Requirements

SR-RES-0010/1.0 T

SF Server shall be deployed on a server running Linux and the Apache Tomcat web application server.

5.7. Design and Implementation Constraints

SR-IMP-0010/1.0 R

The SFRE development process shall be based on [AD 9].

SR-IMP-0020/1.0 R

The SFRE requirement analysis shall be performed using UML, including the production of use cases and interaction diagrams for the various message types

SR-IMP-0030/1.0 R

The SFRE design shall be performed using UML

5.8. Software Configuration and Delivery Requirements

SR-CON-0010/1.0 R

The configuration aspects applicable for the project shall be defined in the Software Configuration Management Plan, included in [AD 7].

SF Server Configuration of *Sensor unavailability*

SF server configuration shall include the modelling of *Sensor unavailability*, i.e. periods where the sensor is identified as unavailable. The model shall include a time when this unavailability becomes known. Requests made after this time that would use the sensor during the unavailable period shall be returned as not feasible. In situations where the plan was requested before this time then the simulation shall eventually have a status indicating that the coverage was not acquired.

SF Server Configuration of *Weather conditions*

SF server configuration shall include the modelling of *Weather conditions*. The model shall have a simple weather forecast, based on the data typically extracted from a Weather Research and Forecasting (WRF) model, as used for actual forecasts. From these there is an estimated percentage chance of the weather being too bad for a successful acquisition within a certain region, for a certain period. The simulation shall work on some simple heuristics based on the proximity of the acquisition time to the current time, and the estimated percentage chance of bad weather to establish if there is any point in planning optical acquisitions, or if the tasking is to be considered not feasible. The percentage chance of weather affecting the acquisition shall also be returned in the server's responses.

SF Server Configuration of *Station unavailability*

SF server configuration shall include the modelling of *Station unavailability*: this shall be modelled in a similar way to Sensors – with periods of unavailability becoming known at certain points in time. The unavailability shall be considered to affect an acquisition if:

1) the station is the next one the satellite would pass after the acquisition, and
2) the unavailability period covers any of the time when the satellite is in view.

5.9. Adaptation and Installation Requirements

SR-INS-0010/1.0 R

For any software components delivered for deployment at ESRIN instructions shall be produced regarding how to install the software. This could include information for the SF Server and SF Client software developed and/or the used libraries and COTS.

5.10. Verification, Validation and Acceptance Requirements

SR-VVA-0010/1.0 R

Unit test shall be designed, implemented an executed to test each individual software component composing the system to be developed. Stubs shall be used in case of component interactions.

SR-VVA-0020/1.0 R

Integration tests shall be designed, implemented and executed to mainly verify the interfaces between software components.

SR-VVA-0030/1.0 R

In order to support the demonstration activities in the Acceptance Review a System Test Plan shall be written and system testing shall be carried out at DEIMOS resulting in Validation Testing Reports. These tests shall be based on, and shall reference, this Software Requirements Document and the SPS EO Application profile [RD 1].

SR-VVA-0040/1.0 R

An Abstract Test Suite shall be specified as part of [RD 1] and an Executable Test Suite will be deployed for use in testing the SF Server.
It will consist of a set of test scripts that shall be delivered to ESA.

6. TRACEABILITY MATRICES

6.1. Direct Traceability

Requirement	RBD Requirement ID(s)	Comment
REQ-FUN-0010/1.0	SR-FUN-0010/1.0	
	SR-FUN-0020/1.0	
	SR-FUN-0030/1.0	
	SR-FUN-0040/1.0	
	SR-FUN-0050/1.0	
	SR-FUN-0060/1.0	
	SR-FUN-0070/1.0	
	SR-FUN-0080/1.0	
	SR-FUN-0090/1.0	
	SR-FUN-0100/1.0	
	SR-FUN-0110/1.0	
	SR-FUN-0120/1.0	
	SR-FUN-0130/1.0	
	SR-FUN-0140/1.0	
	SR-FUN-0150/1.0	
	SR-FUN-0180/1.0	
	SR-FUN-0230/1.0	
	SR-FUN-0240/1.0	
	SR-FUN-0250/1.0	
	SR-FUN-0410/1.0	
	SR-FUN-0810/1.0	
	SR-FUN-0940/1.0	
	SR-FUN-0960/1.0	
	SR-FUN-0980/1.0	
	SR-FUN-1010/1.0	
	SR-FUN-1030/1.0	
	SR-FUN-1050/1.0	

REQ-FUN-0020/1.0	SR-FUN-0050/1.0	
REQ-FUN-0030/1.0	SR-FUN-0060/1.0	
REQ-FUN-0040/1.0	SR-FUN-0060/1.0	
REQ-FUN-0050/1.0	SR-FUN-0220/1.0	
REQ-FUN-0060/1.0	SR-FUN-0200/1.0 SR-FUN-0210/1.0 SR-FUN-0230/1.0 SR-FUN-0970/1.0	
REQ-FUN-0070/1.0	SR-FUN-0990/1.0	
REQ-FUN-0080/1.0	SR-FUN-0200/1.0 SR-FUN-0220/1.0 SR-FUN-0970/1.0	
REQ-FUN-0090/1.0	SR-FUN-0920/1.0 SR-FUN-0930/1.0 SR-FUN-0980/1.0 SR-FUN-1000/1.0	
REQ-FUN-0100/1.0	SR-FUN-0950/1.0	
REQ-FUN-0110/1.0	SR-FUN-0080/1.0	
REQ-FUN-0120/1.0	SR-FUN-0160/1.0 SR-FUN-0170/1.0 SR-FUN-0830/1.0 SR-FUN-0840/1.0 SR-CON-0010/1.0	
REQ-FUN-0130/1.0	SR-FUN-0190/1.0 SR-FUN-0430/1.0 SR-FUN-0510/1.0 SR-FUN-0520/1.0 SR-FUN-0530/1.0	
REQ-PER-0010/1.0	SR-PER-0010/1.0 SR-RES-0010/1.0	
REQ-PER-0020/1.0	SR-PER-0020/1.0	
REQ-INT-0010/1.0	SR-FUN-0940/1.0	

REQ-INT-0020/1.0	SR-FUN-0320/1.0 SR-FUN-0610/1.0 SR-FUN-0620/1.0 SR-INT-0020/1.0	

6.2. Inverse Traceability

Requirement	RBD Requirement ID(s)	Comment
SR-FUN-0010/1.0	REQ-FUN-0010/1.0	
SR-FUN-0020/1.0	REQ-FUN-0010/1.0	
SR-FUN-0030/1.0	REQ-FUN-0010/1.0	
SR-FUN-0040/1.0	REQ-FUN-0010/1.0	
SR-FUN-0050/1.0	REQ-FUN-0010/1.0 REQ-FUN-0020/1.0	
SR-FUN-0060/1.0	REQ-FUN-0010/1.0 REQ-FUN-0030/1.0 REQ-FUN-0040/1.0	
SR-FUN-0070/1.0	REQ-FUN-0010/1.0	
SR-FUN-0080/1.0	REQ-FUN-0010/1.0 REQ-FUN-0110/1.0	
SR-FUN-0090/1.0	REQ-FUN-0010/1.0	
SR-FUN-0100/1.0	REQ-FUN-0010/1.0	
SR-FUN-0110/1.0	REQ-FUN-0010/1.0	
SR-FUN-0120/1.0	REQ-FUN-0010/1.0	
SR-FUN-0130/1.0	REQ-FUN-0010/1.0	
SR-FUN-0140/1.0	REQ-FUN-0010/1.0	
SR-FUN-0150/1.0	REQ-FUN-0010/1.0	
SR-FUN-0160/1.0	REQ-FUN-0120/1.0	
SR-FUN-0170/1.0	REQ-FUN-0120/1.0	
SR-FUN-0180/1.0	REQ-FUN-0010/1.0	
SR-FUN-0190/1.0	REQ-FUN-0130/1.0	

SR-FUN-0200/1.0	REQ-FUN-0060/1.0 REQ-FUN-0080/1.0	
SR-FUN-0210/1.0	REQ-FUN-0060/1.0	
SR-FUN-0220/1.0	REQ-FUN-0050/1.0 REQ-FUN-0080/1.0	
SR-FUN-0230/1.0	REQ-FUN-0010/1.0 REQ-FUN-0060/1.0	
SR-FUN-0240/1.0	REQ-FUN-0010/1.0	
SR-FUN-0250/1.0	REQ-FUN-0010/1.0	
SR-FUN-0310/1.0		Software requeriment; not derived from a baseline requeriment.
SR-FUN-0320/1.0	REQ-INT-0020/1.0	
SR-FUN-0410/1.0	REQ-FUN-0010/1.0	
SR-FUN-0420/1.0		Software requeriment; not derived from a baseline requeriment.
SR-FUN-0430/1.0	REQ-FUN-0130/1.0	
SR-FUN-0510/1.0	REQ-FUN-0130/1.0	
SR-FUN-0520/1.0	REQ-FUN-0130/1.0	
SR-FUN-0530/1.0	REQ-FUN-0130/1.0	
SR-FUN-0610/1.0	REQ-INT-0020/1.0	
SR-FUN-0620/1.0	REQ-INT-0020/1.0	
SR-FUN-0710/1.0		Implementation constraint; not derived from a baseline requeriment.
SR-FUN-0720/1.0		Implementation constraint; not derived from a baseline requeriment.
SR-FUN-0810/1.0	REQ-FUN-0010/1.0	
SR-FUN-0820/1.0		Implementation constraint; not derived from a baseline requeriment.
SR-FUN-0840/1.0	REQ-FUN-0120/1.0	
SR-FUN-0910/1.0		Implementation constraint; not derived from a baseline requeriment.
SR-FUN-0920/1.0	REQ-FUN-0090/1.0	Execution constraint; not derived from a baseline requeriment.
SR-FUN-0930/1.0	REQ-FUN-0090/1.0	Software requeriment; not derived from a

		baseline requeriment.
SR-FUN-0940/1.0	REQ-FUN-0010/1.0 REQ-INT-0010/1.0	
SR-FUN-0950/1.0	REQ-FUN-0100/1.0	
SR-FUN-0960/1.0	REQ-FUN-0010/1.0	
SR-FUN-0970/1.0	REQ-FUN-0060/1.0 REQ-FUN-0080/1.0	
SR-FUN-0980/1.0	REQ-FUN-0010/1.0 REQ-FUN-0090/1.0	
SR-FUN-0990/1.0	REQ-FUN-0070/1.0	
SR-FUN-1000/1.0	REQ-FUN-0090/1.0	
SR-FUN-1010/1.0	REQ-FUN-0010/1.0	
SR-FUN-1030/1.0	REQ-FUN-0010/1.0	
SR-FUN-1050/1.0	REQ-FUN-0010/1.0	
SR-PER-0010/1.0	REQ-PER-0010/1.0	
SR-PER-0020/1.0	REQ-PER-0020/1.0	
SR-INT-0010/1.0		Deployment requeriment; not derived from a baseline requeriment.
SR-INT-0020/1.0	REQ-INT-0020/1.0	
SR-OPE-0010/1.0		Operational requirement; not derived from a baseline requirement.
SR-RES-0010/1.0	REQ-PER-0010/1.0	
SR-IMP-0010/1.0		Design constraint; not derived from a baseline requeriment.
SR-IMP-0020/1.0		Design constraint; not derived from a baseline requeriment.
SR-IMP-0030/1.0		Design constraint; not derived from a baseline requeriment.
SR-CON-0010/1.0	REQ-FUN-0120/1.0	
SR-INS-0010/1.0		Installation requirement; not derived from a baseline requirement.
SR-VVA-0010/1.0		Validation requirement; not derived from a baseline requirement.
SR-VVA-0020/1.0		Validation requirement; not derived from a

		baseline requirement.
SR-VVA-0030/1.0		Validation requirement; not derived from a baseline requirement.
SR-VVA-0040/1.0		Validation requirement; not derived from a baseline requirement.