Open Geospatial Consortium Inc.

Date: 2007-05-10

Reference number of this document: **OGC 07-018**

Version: 0.9.4

Category: OpenGIS® Discussion Paper

Editor: Philippe Mérigot, Spot Image

OpenGIS® Sensor Planning Service Application Profile for EO Sensors

Copyright

Warning

.

Document type:	OpenGIS® Discussion Paper
Document subtype:	**SPS Application Schema**
Document stage:	Draft proposed version
Document language:	English

Contents

Page

List of figures

i. Preface

This Discussion Paper explains how a Sensor Planning Service is organised and implemented for the Earth Observation domain.

The final goal being to agree to a coherent set of interfaces for sending a programming request 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.

This document has used the Sensor Planning Service (SPS) [OGC 05-089r3] Candidate Implementation Specification as input.

ii. Document terms and definitions

This document uses the specification terms defined in Subclause 5.3 of [OGC 05-008], which is based on the ISO/IEC Directives, Part 2, Rules for the structure and drafting of International Standards.

iii. Submitting organizations

The following organisations will submit the original document or its revisions to the OGC® SPS Revision Working Group.

- **ESA – European Space Agency**
- **Spacebel s.a.**
- **Astrium**
- **Spot Image**

The editors would like to acknowledge that this work is the result of collaboration and review of many organizations and would like to thank for the comments and contributions from:

- **ASI**
- **CNES**
- **DLR**
- **Eumetsat**
- **MDA**
- **EUSC**

Note: this does not imply a complete endorsement from these organizations.

iv. Document contributor contact points

All questions regarding this document should be directed to the editor. Additional contributors are listed below:

Name	Organization	Contribution	Email
Didier Giacobbo	Spot Image	Initial version	didier.giacobbo [at] spotimage.fr
JC Angulo	Spot Image	List of input parameters	jean-christophe.angul [at] spotimage.fr
Daniele Marchionni	DATAMAT	OR11, Review and comments	Daniele.marchionni [at] datamat.it
Jolyon Martin	ESA	Review and comments	Jolyon.Martin [at] esa.int
Monique Benhamou	Astrium		monique.benhamou [at] astrium.eads.net
Yves Coene	Spacebel		Yves.coene [at] spacebel.be
Patrick Floissac	Magellium (CNES sub contractor)	Review and comments	Patrick.floissac [at] magellium.cnes.fr
Ingo Simonis	Geospatial Research & Consulting	Review and comments	ingo.simonis [at] geospatialresearch.de
Alexandre Robin	Sensia Software	SensorML examples	robin.alexandre [at] gmail.com

v. Revision history

Date	Release	Editor	Primary clauses modified	Description
2006-07-28	0.0.1	Didier Giacobbo	initial version	initial version;
2006-09-04	0.0.2	Didier Giacobbo	Small update	Description of Simulated Sensor Workload and parameters to delegated mission plan added
2006-10-26	0.0.3	Didier Giacobbo	Major update	Add of new operations: DelegatedMissionPlan, Update Update of the XML example Add of UML description for the EO aspects
2006-11-22	0.0.4	Philippe Mérigot	Major update	DescribeTasking replaced by DescribeGetFeasibility and DescribeSubmit. Previous DescribeGetFeasibility removed. xxxRequestResponse renamed in xxxResponse Schemas modified: DescribeGetFeasibility, GetFeasibility, GetStatus
2006-12-21	0.9	Didier Giacobbo	Major update	HMA-IF-DAT-MP-0001_v1.0.3 merging
2007-01-12	0.9.1	Didier Giacobbo	Minor update	Editing correction
2007-01-16	0.9.1	Philippe Mérigot	Major update	viii Open issues Future work External interface Preliminary list of Tasking Parameters ordering parameters removed GetCapabilities protocol/encoding + capabilities schema Operations removed: UpdateStatus Operations modified: GetStatus, Cancel, DescribeGetFeasibility, DescribeSubmit XML examples modified: GetFeasibility & Submit request/response, sweCommon instance (annexe)
2007-02-07	0.9.2	Philippe Mérigot	Major update	Definition of acknowledgment messages for asynchronous operations Input parameters:

				• QOS and Priority in a new element *Priority* • AcquisitionMode possible values (OHR) • ValidationParameters • SurveyPeriods Op. modified: DescribeSubmit, Submit & GetStatus
2007-03-21	0.9.3	Philippe Mérigot		Editing correction Future work and open issues Use of UML for the description of the preliminary list of input Parameters GetCapabilities, DescribeSensor, GetFeasibility (feasibility levels) operations modified DelegatedMissionPlan removed Delivery information removed Schemas (annex A), SensorML examples (annex B)
2007-05-10	0.9.4	Carl Reed	Various	Preparation for posting as DP.

vi. Changes to the OGC Abstract Specification

The OpenGIS® Abstract Specification does not require changes to accommodate the technical contents of this document.

vii. Future work

In future versions of the document the following issues will be improved:

- In this document operations such as EstimateSensorWorkload, DescribeGetFeasibility and DescribeSensor are not part of the SPS reference specification [OGC 05-089]. These operations are proposed only for this profile in order to fulfil the EO Sensor Planning Service requirements. Their adoption in the [OGC 05-089] document has to follow the OGC rules.

- The parameters of the EstimateSensorWorkload operation have to be defined precisely. A complete schema applicable to this specification has to be provided.

- The DescribeSensor operation returns two types of document (a complete and a brief description of the sensors), corresponding to two SensorML profiles. These profiles must be specified. The schema of the DescribeSensor response is also missing.

- Regarding ongoing revision work on the SPS specification, this current EO profile uses the SPS specification as is at the time of the OWS4 initiative. Once an official version of the new SPS will be available (version 1.1), the SPS EO Profile will be updated in order to describe only the new operations and parameters.

- The user should be able to validate an acquisition in order to close the loop and update consequently the submitted task. This could be done via the Update operation. For example, the following schema of request Update would allow the client to update a submitted task or validate the acquisition:

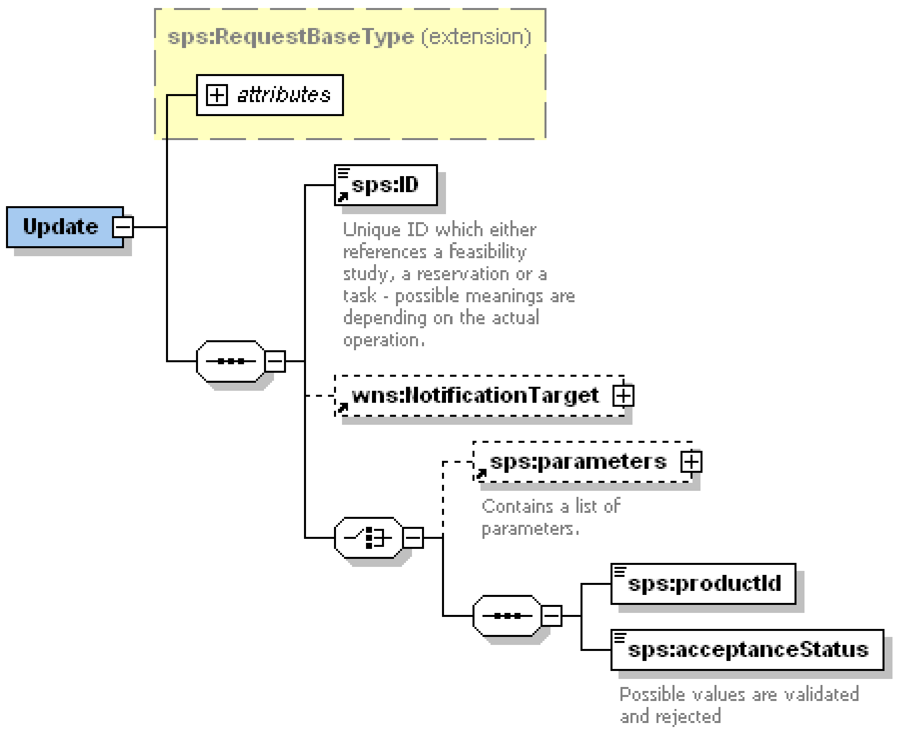

sps:RequestBaseType (extension)
attributes
Update
sps:ID
Unique ID which either references a feasibility study, a reservation or a task - possible meanings are depending on the actual operation.
wns:NotificationTarget
sps:parameters
Contains a list of parameters.
sps:productId
sps:acceptanceStatus
Possible values are validated and rejected

Figure 1-1 - Update request schema

- Request from Datamat: the user may want to get the list of all orders that have been updated since a specified date. The GetStatus operation schemas could be modified in order to satisfy this need : by unbounding the number of ProgressReport elements in the response and making ID optional in GetStatus request, and explaining that if it is not specified it means that all orders issued after the optional DateFrom date have to be returned.

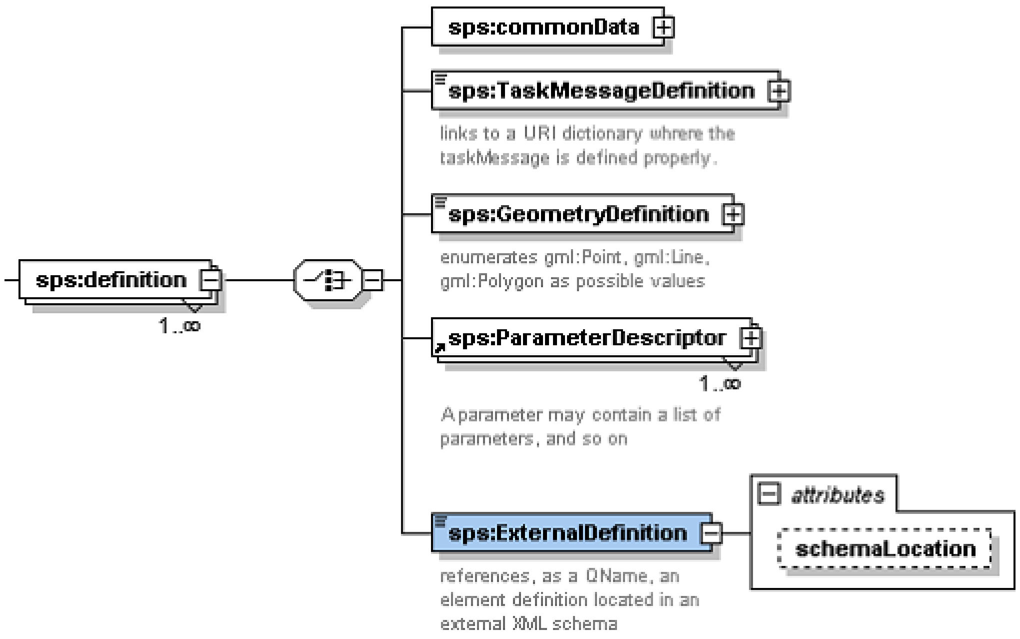
- Request from CNES: from the HMA perspective, data returned by the SPS EO profile is very similar to data returned by a catalog implementing the CSW EO profile : the basic elements are essentially, in both cases, EO products having a footprint, a range of date and some additional properties.
  We expect that, in most cases, HMA-enabled clients will interact with both SPS and CSW instances and will present to the user information mixing acquired datasets with planned or foreseen EO products.

- A GML model has been yet defined for acquired EO products : [OGC 07-018] "GML 3.1.1 Application schema for Earth Observation products".
  This model (or a similar one adapted to the mission planning context) cannot be currently used in the SPS EO profile : the later mandates the use of swe common constructs (with a few GML exceptions as gml:Polygon) whereas our model involves a GML application schema. HMA-enabled clients will have, thus, to handle two very different models.
  We think that this issue is not specific to the HMA project : other communities might, in the future, use SPS with alternative formats and, in one sense, the SPS specification currently lacks of some "extensibility" mechanism.

  To achieve the use of a GML application schema for the HMA community, we suggest to:
  - extend the [OGC 07-018] specification to handle also planned or foreseen EO products.

- provide in the base SPS specification a mean to describe a parameter as “with external format” (eg : specifiying a QName and the associated XML schema). The base specification could restrict the use of such a definition to output parameters and to well-defined communities (i.e. defining SPS profiles).

Suggested change to <sps:definition> in <sps:ParameterDescriptor> is below :


sps:definition
1..∞
sps:commonData
sps:TaskMessageDefinition
links to a URI dictionary whrere the taskMessage is defined properly.
sps:GeometryDefinition
enumerates gml:Point, gml:Line, gml:Polygon as possible values
sps:ParameterDescriptor
1..∞
A parameter may contain a list of parameters, and so on
sps:ExternalDefinition
references, as a QName, an element definition located in an external XML schema
attributes
schemaLocation

Note that the only sps structure to be changed would be the sps:ParameterDescriptor : the sps:Parameter (i.e. the structure embedding the response) can yet vehiculate any type of content.

viii. Open issues

- The preliminary list of input parameters (§ 8.2) may be described by using a dictionnary. In this case, the parameters may be referenced with a URI.

ix. Foreword

This document is an Earth Observation candidate profile of the OGC Implementation Specification for the Sensor Planning Service (SPS) version 0.0.30 [OGC 05-089r3 and 04-092r4].

This document references several external standards and specifications as dependencies:

a) Unified Modeling Language (UML) Version 1.3, The Object Management Group (OMG): http://www.omg.org/cgi-bin/doc?formal/00-03-01

b) The Extensible Markup Language (XML), World Wide Web Consortium, http://www.w3.org/TR/1998/REC-xml-19980210

c) W3C Recommendation (24 June 2003): SOAP Version 1.2 Part 1,Messaging Framework, http://www.w3.org/TR/SOAP/

d) WSDL, Web Services Description Language (WSDL) 1.1, http://www.w3.org/TR/wsdl

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The OGC shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the specification set forth in this document, and to provide supporting documentation.

Introduction

The SPS configuration proposed in this profile is intended to support the programming process of Earth Observation (EO) sensors system. This profile describes a consistent SPS configuration that can be supported by many satellite data providers, most of whom have existing facilities for the management of these programming requests.

The Sensor Planning Service (SPS) is intended to provide a standard interface to collection assets (i.e., sensors, and other information gathering assets) and to the support systems that surround them. Not only must different kinds of assets with differing capabilities be supported, but also different kinds of request processing systems, which may or may not provide access to the different stages of planning, scheduling, tasking, collection, processing, archiving, and distribution of requests and the resulting observation data and information that is the result of the requests. The SPS is designed to be flexible enough to handle such a wide variety of configurations.

OpenGIS® Sensor Planning Service Application Profile for EO Sensors

1 Scope

This SPS EO profile document specifies at a lower level the interfaces and parameters for requesting information describing the capabilities of a Sensor Planning Service dedicated to the EO Sensor domain, for determining the feasibility of an intended sensor planning request, for submitting such a request, for inquiring about the status of such a request, for updating or cancelling such a request, and for requesting information about further OGC Web services that provide access to the data collected by the requested task.

This profile document re-uses largely information models, descriptions and information comprise as defined in within the SPS implementation specification [OGC 05-089 SPS].

This document describes the interfaces for programming the activities of Earth Observation sensors. In particular this candidate implementation specification defines operations for:

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

2 Conformance

Conformance will be tested by the HMA-T project.

3 Normative references

The following normative documents contain provisions that, through reference in this text, constitute provisions of this document. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. For undated references, the latest edition of the normative document referred to applies.

[NR1]	W3C Recommendation January 1999, Namespaces In XML, http://www.w3.org/TR/2000/REC-xml-names.
[NR2]	W3C Recommendation 6 October 2000, Extensible Markup Language (XML) 1.0 (Second Edition), http://www.w3.org/TR/REC-xml
[NR3]	W3C Recommendation 2 May 2001: XML Schema Part 0: Primer, http://www.w3.org/TR/2001/REC-xmlschema-0-20010502/
[NR4]	W3C Recommendation 2 May 2001: XML Schema Part 1: Structures, http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/
[NR5]	W3C Recommendation 2 May 2001: XML Schema Part 2: Datatypes, http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/
[NR6]	W3C Recommendation (24 June 2003): SOAP Version 1.2 Part 1: Messaging Framework, http://www.w3.org/TR/SOAP/
[NR7]	WSDL, Web Services Description Language (WSDL) 1.1. Available [online]: http://www.w3.org/TR/wsdl
[NR9]	OGC 05-008c1 OWS Common Implementation Specification, May 2005
[NR11]	OGC 06-080 GML 3.1.1 Application schema for Earth Observation products
[NR12]	OGC 06-141 r2 Ordering Services for Earth Observation Products
[NR13]	OGC-05-089r3 OpenGIS® Sensor Planning Service Implementation Specification
[NR14]	OGC 05-086 SensorML

3.1 Other references

[OR1]	HMA-PL-SPB-AV-001 HMA Prototype Acceptance Test Plan
[OR2]	OGC-05-057r4 OpenGIS Catalogue Services Best Practices for EO Products
[OR3]	OGC 04-038r4 OpenGIS® Catalogue Services Specification 2.0.1 (with Corrigendum) ISO Metadata Application Profile
[OR4]	OpenGIS® Sensor Planning Service Application Profile for EO Sensors
[OR7]	OGC 05-090 SWE Architecture
[OR8]	ISO 19105:2000 *Geographic information — Conformance and Testing*
[OR10]	COMU-TS-ASU-RB-008
[OR11]	HMA-IF-DAT-MP-0001_v1.0.3 Programming Services for Earth Observation Products D. Marchionni, DATAMAT spa.

In addition to this document, this specification includes several normative XML Schema documents as specified in Annex A.

4 Terms and definitions

For the purposes of this specification, the definitions specified in Clause 4 of the OWS Common Implementation Specification [OGC 05-008] shall apply. In addition, the following terms and definitions apply.

4.1 Application profile
set of one or more base standards and – where applicable – the identification of chosen clauses, classes, subsets, options and parameters of those base standards that are necessary for accomplishing a particular function [ISO 19101, ISO 19106]

4.2 asset
synonyms: sensor, simulation
an available means. For the SPS, an available means of collecting information.

4.3 asset management system
Synonyms: acquisition system, asset support system
A system for controlling the effective utilization of an asset

4.4 client
software component that can invoke an **operation** from a **server**

4.5 collection
Process sense (default for this document): the act of gathering something together
Result sense: an aggregation of the results of one or more collection processes.

4.6 data clearinghouse

collection of institutions providing digital data, which can be searched through a single interface using a common metadata standard [ISO 19115]

4.7 data level

stratum within a set of layered levels in which data is recorded that conforms to definitions of types found at the application model level [ISO 19101]

4.8 dataset series (dataset collection[1])

collection of datasets sharing the same product specification [ISO 19113, ISO 19114, ISO 19115]. In this context, a collection metadata record in the catalogue describes a collection of EO Products, typically a dataset collection corresponds to datasets (i.e. products) generated by a single sensor in a specific mode on a particular EO satellite.

4.9 geographic dataset

dataset with a spatial aspect [ISO 19115]

4.10 geographic information

information concerning phenomena implicitly or explicitly associated with a location relative to the Earth [ISO 19128 draft]

4.11 georesource

geographic information of a specific type (e.g. geographic dataset, geographic application, geographic service)

4.12 identifier

a character string that may be composed of numbers and characters that is exchanged between the client and the server with respect to a specific identity of a resource

4.13 interface

named set of operations that characterise the behaviour of an entity [ISO 19119]

4.14 metadata dataset (metadataset)

metadata describing a specific dataset [ISO 19101]

4.15 metadata entity

group of metadata elements and other metadata entities describing the same aspect of data

NOTE 1 A metadata entity may contain one or more metadata entities.

NOTE 2 A metadata entity is equivalent to a class in UML terminology [ISO 19115].

4.16 metadata schema

conceptual schema describing metadata

NOTE ISO 19115 describes a standard for a metadata schema. [ISO 19101]

4.17 metadata section

subset of metadata that defines a collection of related metadata entities and elements [ISO 19115]

[1] Due to historical reasons we´ll mainly use the term ‘dataset collection’ in this document although the term ‘dataset series’ is used in the ISO/TC211 Terminology Maintenance Group.

4.18 operation
specification of a transformation or query that an object may be called to execute [ISO 19119]

4.19 parameter
variable whose name and value are included in an operation **request** or **response**

4.20 profile
set of one or more base standards and – where applicable – the identification of chosen clauses, classes, subsets, options and parameters of those base standards that are necessary for accomplishing a particular function [ISO 19101, ISO 19106]

4.21 qualified name
name that is prefixed with its naming contex**t**

4.22 request
invocation of an **operation** by a **client**

4.23 requirement
Something that is necessary in advance

4.24 response
result of an **operation,** returned from a **server** to a **client**

4.25 schema
formal description of a model [ISO 19101, ISO 19103, ISO 19109, ISO 19118]

4.26 server
service instance
a particular instance of a **service** [ISO 19119]

4.27 service
distinct part of the functionality that is provided by an entity through interfaces [ISO 19119]

capability which a service provider entity makes available to a service user entity at the interface between those entities [ISO 19104 terms repository]

4.28 service interface
shared boundary between an automated system or human being and another automated system or human being [ISO 19101]

4.29 service metadata
metadata describing the **operations** and **geographic information** available at a **server** [ISO 19128 draft]

4.30 state
condition that persists for a period

NOTE The value of a particular feature attribute describes a condition of the feature [ISO 19108].

4.31 transfer protocol
common set of rules for defining interactions between distributed systems [ISO 19118]

4.32 version

version of an Implementation Specification (document) and XML Schemas to which the requested operation conforms

NOTE An OWS Implementation Specification version may specify XML Schemas against which an XML encoded operation request or response must conform and should be validated.

5 Symbols and abbreviations

5.1 Symbols (and abbreviated terms)

Some frequently used abbreviated terms:

API	Application Program Interface
ATM	Atmospheric
COTS	Commercial Off The Shelf
CQL	Common Query Language
CRS	Coordinate Reference System
CSW	Catalogue Service-Web
DCE	Distributed Computing Environment
DC	Dublin Core
DCMI	Dublin Core Metadata Initiative
DCP	Distributed Computing Platform
EO	Earth Observation
HMA	Heterogeneous Missions Accessibility
HTTP	HyperText Transport Protocol
ISO	International Organisation for Standardisation
OGC	Open GIS Consortium
OHR	Optical High Resolution
SAR	Synthetic Aperture Radar
SOAP	Simple Object Access Protocol
SPS	Sensor Planning Service
SQL	Structured Query Language
UML	Unified Modeling Language

URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URN	Uniform Resource Name
UTF-8	Unicode Transformation Format-8
WSDL	Web Service Definition Language
W3C	World Wide Web Consortium
XML	eXtensible Markup Language

5.2 UML notation

5.2.1 UML Class Diagrams

Some of the diagrams in this document are presented using the Unified Modeling Language (UML) static structure diagram. The UML notations used in this document are described in Figure 3-1, below.

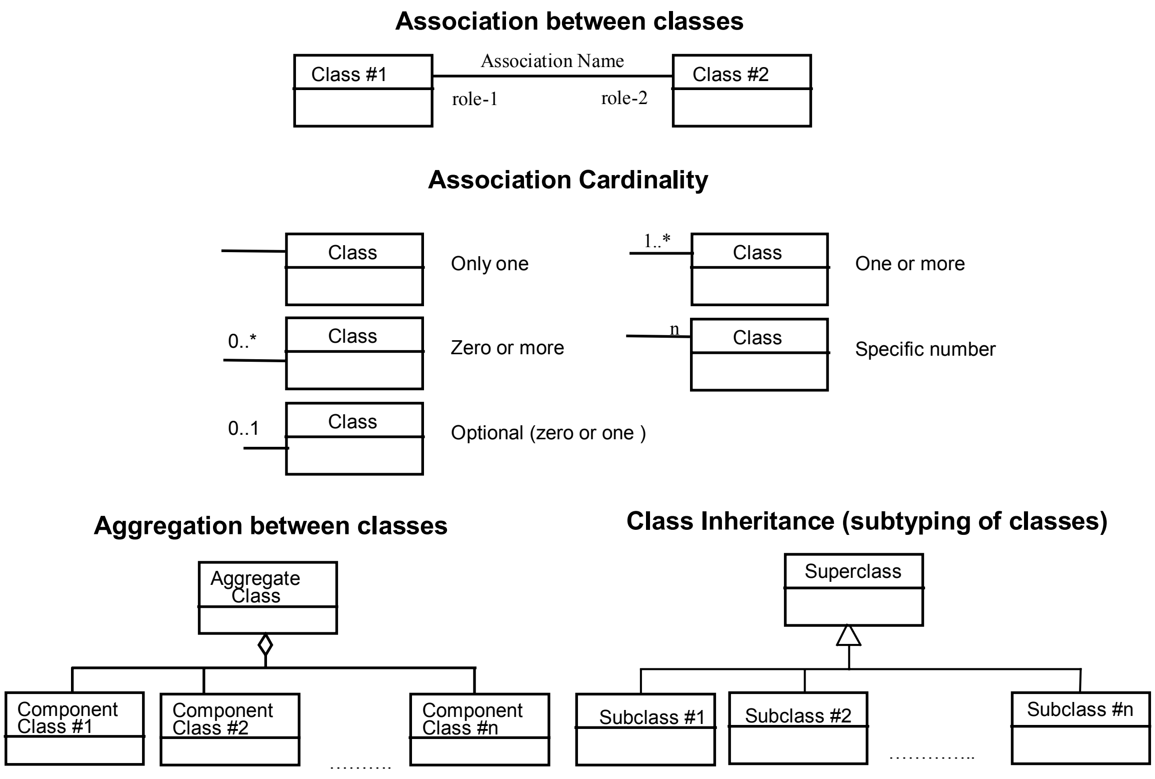


Figure 3-1 UML notations

In these UML class diagrams, the class boxes with a light background are the primary classes being shown in this diagram, often the classes from one UML package. The class boxes with a grey background are other classes used by these primary classes, usually classes from other packages.

In this diagram, the following stereotypes of UML classes are used:

<<Interface>> A definition of a set of operations that is supported by objects having this interface. An Interface class cannot contain any attributes.

<<Type>> A stereotyped class used for specification of a domain of instances (objects), together with the operations applicable to the objects. A Type class may have attributes and associations.

<<DataType>> A descriptor of a set of values that lack identity (independent existence and the possibility of side effects). A DataType is a class with no operations whose primary purpose is to hold the information.

<<CodeList>> A flexible enumeration that uses string values for expressing a list of potential values. If the list alternatives are completely known, an enumeration shall be used; if the only likely alternatives are known, a code list shall be used.

<<Enumeration>> A data type whose instances form a list of alternative literal values. Enumeration means a short list of well-understood potential values within a class.

In this document, the following standard data types are used:

CharacterString – A sequence of characters

Boolean – A value specifying TRUE or FALSE

Integer – An integer number

Identifier – Unique identifier of an object

URI – An identifier of a resource that provides more information

URL – An identifier of an on-line resource that can be electronically accessed

5.2.2 UML Sequence Diagrams

Sequence diagrams are a representation of an interaction between objects. A sequence diagram traces the execution of an interaction in time.

The picture below illustrates a sequence diagram.

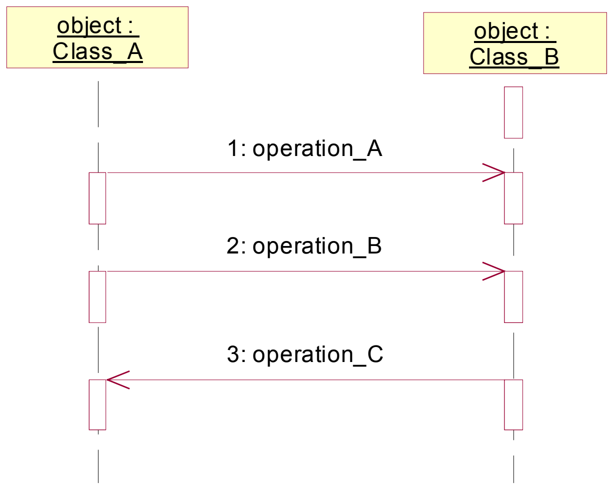


Figure 3-2: UML Sequence Diagrams Notations.

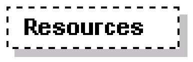
Each interaction between objects is the activation of an operation of an object, which includes input and output parameters.

5.3 XML notation

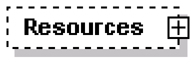
Most diagrams that appear in this specification are presented using an XML schema notation defined by the XMLSpy tool and described in this subclause.

Hereafter the symbols defined in the XML schema notation are described:

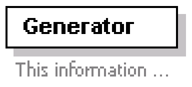
- Optional single element without child elements



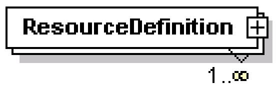
- Optional single element with child elements



- Mandatory single element.



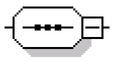
- Mandatory multiple element containing child elements. This element must occur at least once (Minimum Occurrence = 1) and may occur as often as desired (Maximum Occurrence = unbounded).



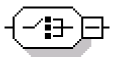
- Mandatory single element with containing simple content (e.g. text) or mixed complex content (e.g. text with xhtml markup).

InternalNotes

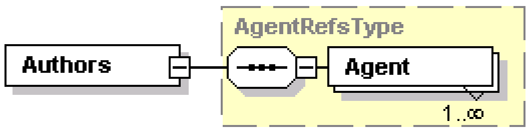
- A sequence of elements. The elements must appear exactly in the sequence in which they appear in the schema diagram.



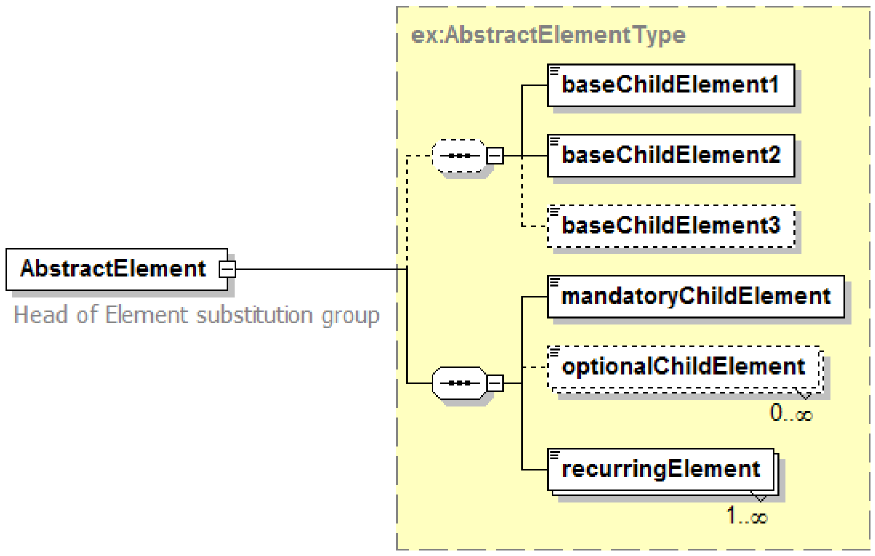
- A choice of elements. Only a single element from those in the choice may appear at this position.



- Types. If an element refers to a complex global type, the type is shown with a border and yellow background.



- Complex Type. The following figure illustrates the use of a complex type for defining an XML element



5.4 Document terms and definitions

This document uses the specification terms defined in Subclause 5.3 of [OGC 05-008c1].

6 System context

This section focuses on the purpose, scope and policies of Programming services that comply with this specification. It documents special requirements and describes the context of use.

6.1 Application domain

The programming service described in this document has the objective of supporting the following 2 types of requests:

- Order of precisely identified (typically specifying the sensing start and stop times) future products. This type of orders are referenced as **Acquisition Orders** in this document;
- Order asking the coverage of a specified area in a specified time window. This type of orders are referenced as **Coverage Orders** in this document;

In this document, these orders will be referred to as **Programming Requests**.

Each requested item in the Programming Request will be referred to as **Task**.

For the first type of programming requests the process is very similar to the one described in [NR12] for ordering products:

- The client identifies (i.e. calculate the sensing start & stop times) the products to be acquired. This step is not covered by this document.
- Next, for each product going to be ordered, the list of tasking parameters is required.
- Next an order for future acquisitions is built on the client selecting the needed tasking parameters for each item to be ordered.
- When the programming request is prepared, the client can ask the feasibility analysis. The result of the analysis can be returned sync / async depending on its complexity, the technical & financial proposal is returned as a document sent by mail / e-mail.
- If the result of the previous step is successful, then the client can submit the programming request. In case of unfeasibility of the request, the service can suggest possible alternative parameters.
- The progress of the programming request can be actively monitored by the client or can be notified to it.
- If necessary the programming request can be cancelled.

For the second type of programming request the process is the same apart from the first step: the client does not have to identify the products, but has to specify only the time window of interest and the geographical area to cover.

6.2 Reference scenarios

This specification refers to 3 scenarios used within the EO domain. The first scenario should be considered as a particular case of the ordering. In this case a request for programming is a “future order”. It is possible to identify uniquely where and when the data will be acquired. This scenario mainly applies to Radar or Atmosphere domains where the weather conditions do not have any influence on the acquisition process. The operations not used in this scenario are DescribeSensor and EstimateSensorWorkload. The response to a GetFeasibility request will be a simple value of type Boolean.

The second reference scenario addresses a more complex request for programming. Here the acquisition needs more than one attempt to acquire the requested data, which comes closer to a "Mission Planning" service scenario. A time range for acquisition has to be defined. In this case the `getFeasibility` operation returns not a simple response of type Boolean anymore. The response should contain either all information necessary to acquire the data or the pseudo scene in the area of interest with a success rate within the acquisition time frame only. The operations not used for such a scenario are `DescribeSensor`, `EstimateSensorWorkload` and `DelegatedMissionPlan`.

The third scenario corresponds to a multi mission level. In this case a request for programming implies more than one mission. All operations described in this specification should be used. Establishing a possible link between multiple missions implies that each mission returns a minimum set of information about the sensor, the workload of the sensor and its capability to reserve a part of the sensor acquisition time.

7 SPS Operations Overview

The SPS operations can be divided into informational and functional operations. The informational operations are the GetCapabilities operation, the DescribeTasking operation, the DescribeResultAccess operation and the GetStatus operation. Other informational operations have to be adopted for the EO profile; these operations are DescribeSensor and DescribeGetFeasibility. Among these, the GetCapabilities, the DescribeResultAccess and the GetStatus operations provide information that the SPS user needs to know, while the DescribeTasking operation provides a description of information that a sensor management system needs to know. The functional operations are the GetFeasibility, the Submit, the Update and Cancel operations. All of these operations have an effect on the sensor management system, as explained below. Another functional operation, EstimateSensorWorkload, has to be adopted for the EO profile.

The SPS EO application profile interface specifies 11 operations that can be requested by a client and performed by a SPS server. Those operations are:

a) GetCapabilities (mandatory) – 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 (mandatory) – This operation allows the client to obtain a description of the sensors supported by the current SPS. The client may request a brief description of the sensor, or a complete one, giving him the capability to simulate the possible acquisition of the sensor.

c) EstimateSensorWorkload (optional) – This operation provides information of the Workload of the called sensor. The description of this workload is under the responsibility of the mission and the freshness of the information will be indicated by the mission.

d) DescribeGetFeasibility (optional) – This operation allows a client to request the information that is needed in order to send a GetFeasibility request. The response contains a description of the input and optionally the output parameters for the GetFeasibility operation. Note: this operation is optional because GetFeasibility is optional.

e) GetFeasibility (optional) – This operation is to provide feedback to a client about the feasibility of a programming request. Dependent on the sensor type façaded 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.

f) DescribeSubmit (mandatory) – This operation allows clients to request the information that is needed in order to send a Submit request. This optional operation should be used only if the input parameters of a submit request are different than the input parameters of a GetFeasibility request.
Note: this operation if mandatory because Submit is mandatory.

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

h) GetStatus (optional) – This operation allows a client to receive information about the current status of the requested task.

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

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

k) DescribeResultAccess (mandatory) – 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 a OGC Web services nevertheless OGC Web services such as SOS, WMS, WFS or WCS are desirable.

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]. Many of these common aspects are normatively referenced herein, instead of being repeated in this specification.

8 Information models for EO Programming Requests

To specify a programming request, the programming parameters have to be specified.

The programming parameters are modelled by the ParameterDescriptor. The names of these parameters shall be aligned with those used in [NR11].

8.1 ParameterDescriptor element

The ParameterDescriptor defines the input a client has to provide to task an asset. This element in the EO profile has been modified as follows:

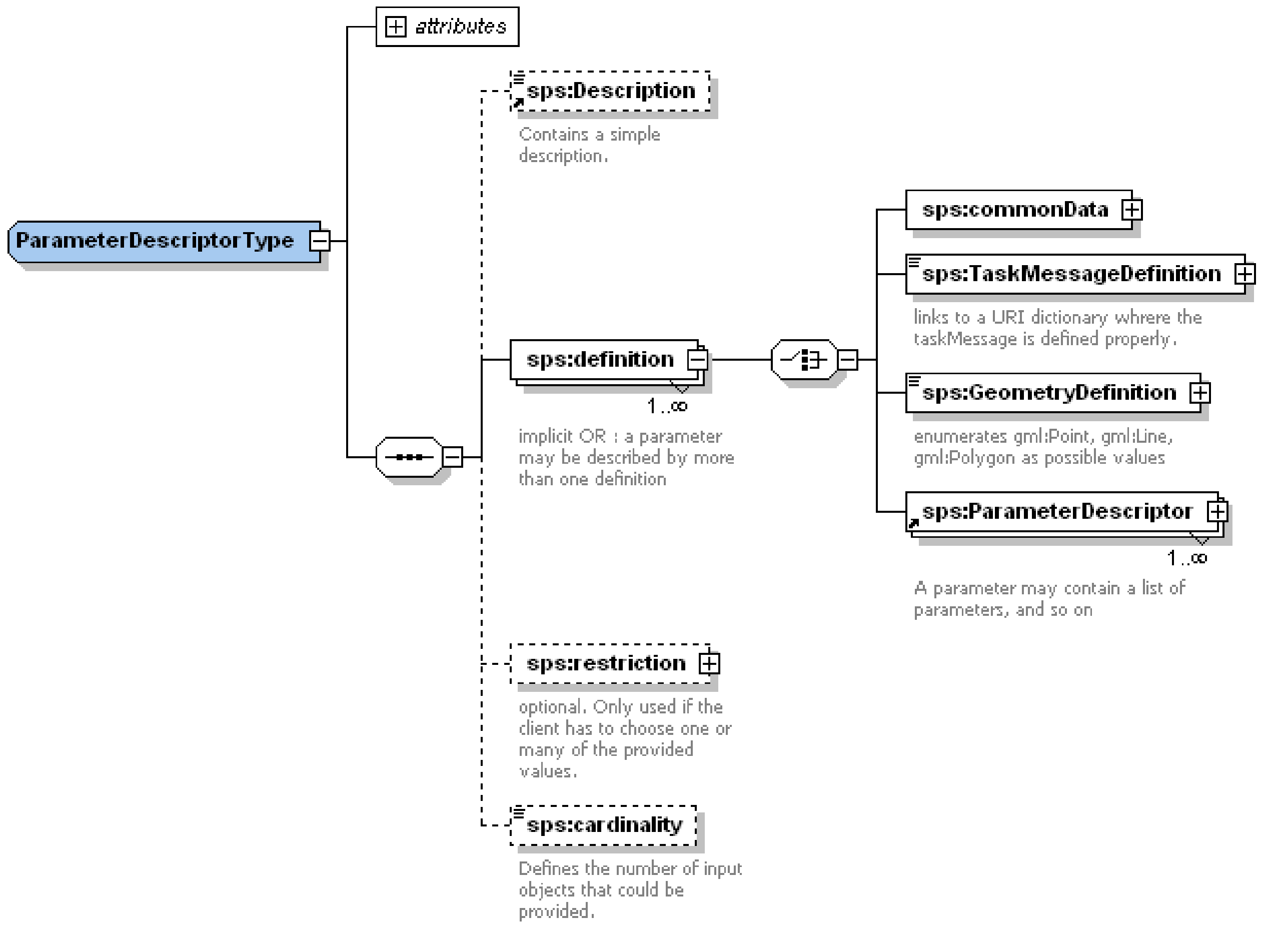

attributes
sps:Description
Contains a simple description.
ParameterDescriptorType
sps:definition
1..∞
implicit OR : a parameter may be described by more than one definition
sps:commonData
sps:TaskMessageDefinition
links to a URI dictionary whrere the taskMessage is defined properly.
sps:GeometryDefinition
enumerates gml:Point, gml:Line, gml:Polygon as possible values
sps:ParameterDescriptor
1..∞
A parameter may contain a list of parameters, and so on
sps:restriction
optional. Only used if the client has to choose one or many of the provided values.
sps:cardinality
Defines the number of input objects that could be provided.

Figure 8-1 – ParameterDescriptor

Changes are:

- **sps:definition** is unbounded (in the SPS specification : minOcc=1 and maxOcc=1) Reason : a parameter may be described by more than one definition. Example: a **regionOfInterest** may be a gml:Polygon OR a Circle

- the choice of the **sps:definition** element contains a **ParameterDescriptorType** element. This allows a parameter to contain a list of parameters and so on. Example: a **scene** parameter contains an ID (String) parameter, a bounding box (gml:Polygon) parameter, etc.

The following example illustrates the use of the **ParameterDescriptor** element:

```
<ParameterDescriptor parameterID="resolution" use="required" updateable="false">
 <Description>Sensor resolution</Description>
 <definition>
  <commonData>
   <swe:Quantity uom="urn:ogc:def:unit:meter">
    <swe:constraint>
     <swe:AllowedValues>
      <swe:valueList>2.5 5 10 20</swe:valueList>
     </swe:AllowedValues>
    </swe:constraint>
   </swe:Quantity>
  </commonData>
 </definition>
 <cardinality>1</cardinality>
</ParameterDescriptor>
```

An example of input parameters for SPOT5 tasking is given in annex B of this document.

8.2 Preliminary List of Tasking Parameters

This paragraph proposes the preliminary and extensible list of parameters a client has to provide to task an EO profile asset. The objective is, in a multi mission context, to define the name and the type of the parameters shared by all the EO missions.

Through the DescribeGetFeasibility and DescribeSubmit operations, each mission has the possibility to:

- mark a parameter as optional or mandatory

- restrict the possible values of a parameter

- add its own specific parameters

Note : the description of the parameters returned by DescribeGetFeasibility and DescribeSubmit operations is given by using the *ParameterDescriptor* element (cf. § 8.1). To make is easier to understand, the description of the parameters is given here by using UML notation and tables.

The parameters are grouped according to their nature:

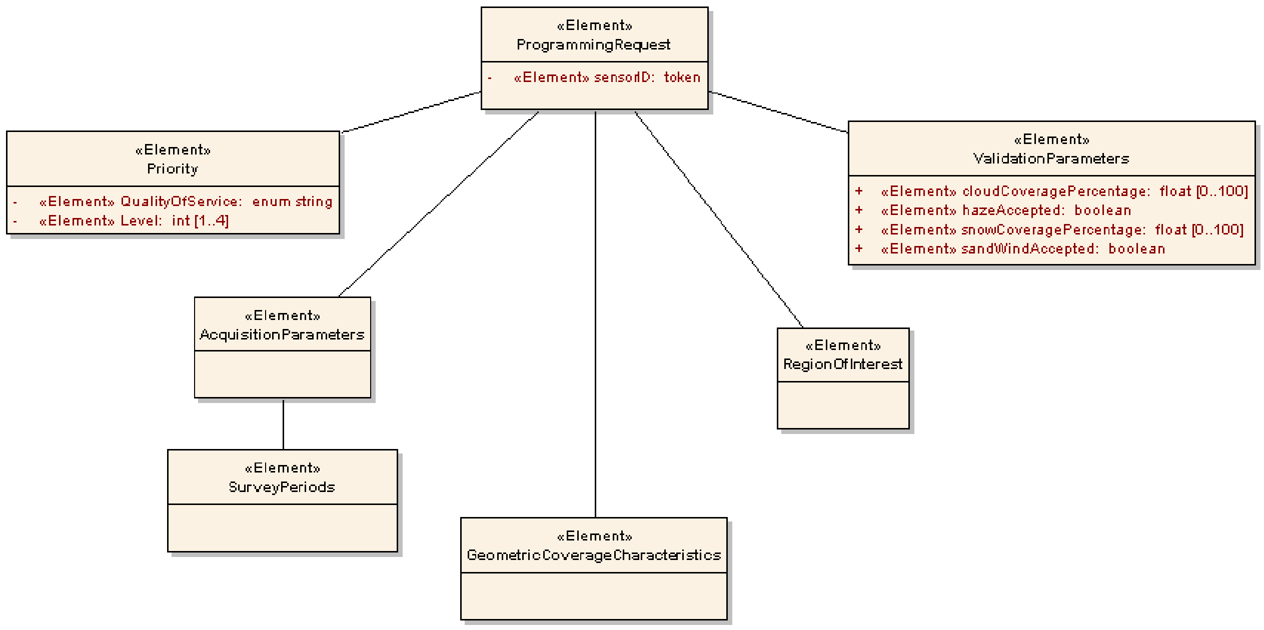

«Element»
ProgrammingRequest
- «Element» sensorID: token
«Element»
Priority
- «Element» QualityOfService: enum string
- «Element» Level: int [1..4]
«Element»
ValidationParameters
+ «Element» cloudCoveragePercentage: float [0..100]
+ «Element» hazeAccepted: boolean
+ «Element» snowCoveragePercentage: float [0..100]
+ «Element» sandWindAccepted: boolean
«Element»
AcquisitionParameters
«Element»
RegionOfInterest
«Element»
SurveyPeriods
«Element»
GeometricCoverageCharacteristics

Figure 8-2 - list of tasking parameters

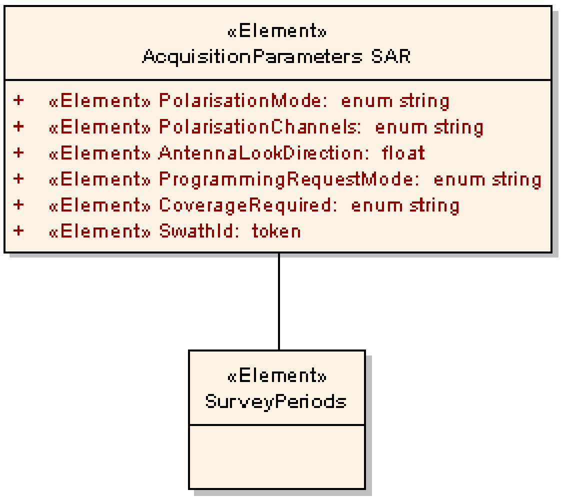
The following tables describe each group and each parameter in details. The first column contains the name of the group or the element; the second column contains the description; the third specifies the source document from which the parameter has been derived; the "Mission" column specifies whether the parameter is applicable for Optical (OHR), Radar (SAR), Atmospheric (ATM) or all.

8.2.1 Priority

Tasking Parameter Name	Description	Source	Mission
Priority	Programming Priority		All
QualityOfService	Quality Of Service Type: Enumerated string Possible values are: Routine, Urgent, Crisis	[OR4]	
Level	Priority level. Type: Integer Possible values are : 1,2,3,4		All

8.2.2 Acquisition parameters

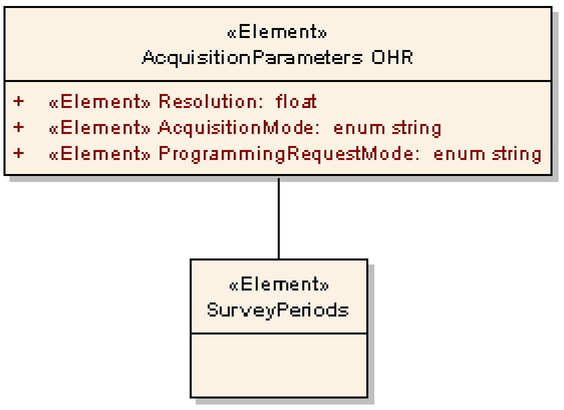
8.2.2.1 SAR mission (radar)


«Element»
AcquisitionParameters SAR
+ «Element» PolarisationMode: enum string
+ «Element» PolarisationChannels: enum string
+ «Element» AntennaLookDirection: float
+ «Element» ProgrammingRequestMode: enum string
+ «Element» CoverageRequired: enum string
+ «Element» SwathId: token
«Element»
SurveyPeriods

Tasking Parameter Name	Description	Source	Mission
PolarisationMode	Polarisation Mode Type: Enumerated String Possible values: D, Q, S, T, UNDEFINED	[NR11]	SAR
PolarisationChannels	Polarisation channel transmit/receive configuration: horizontal, vertical. Type: Enumerated string Possible values: • HH • VH • VV • HH, VV • HH, VH • HH, HV • HV, VV • VH, VV • VH, HV • VV, HH • HH, VV, HV, VH UNDEFINED	[NR11]	SAR
AntennaLookDirection	Antenna look direction Type: floating point Unit: degrees	[NR11]	SAR
ProgrammingRequestMode	Programming type. Type: Enumerated string Possible values are: • SCENE, for tasks specified by time intervals / orbit segments • COVERAGE, for specified coverage orders	[OR4]	SAR

Tasking Parameter Name	Description	Source	Mission
CoverageRequired	Required coverage for the observation Type: Enumerated string Valid values: **"ANY_COVERAGE"**:acquire any product visible in the area of interest (regionOfInterest parameter) even if overlapping already required sub-areas; **"ANY_REFERENCE_COVERAGE"**: acquire any product visible in the area of interest (regionOfInterest parameter), but suppress duplicates of the same relative segment (i.e. same relative orbit and passCoverage); **"FULL_COVERAGE"**: within the period (startDate & completionDate) the area (regionOfInterest parameter) has to be fully mapped.		SAR
SwathId	Indication of a specific swath achieved by steering In case the sensor viewing characteristics can be changed by steering the instrument the SwathId identifies the specific Swath used for the single acquisition Type: String	[NR11]	SAR
SurveyPeriods	cf § 8.2.6		SAR

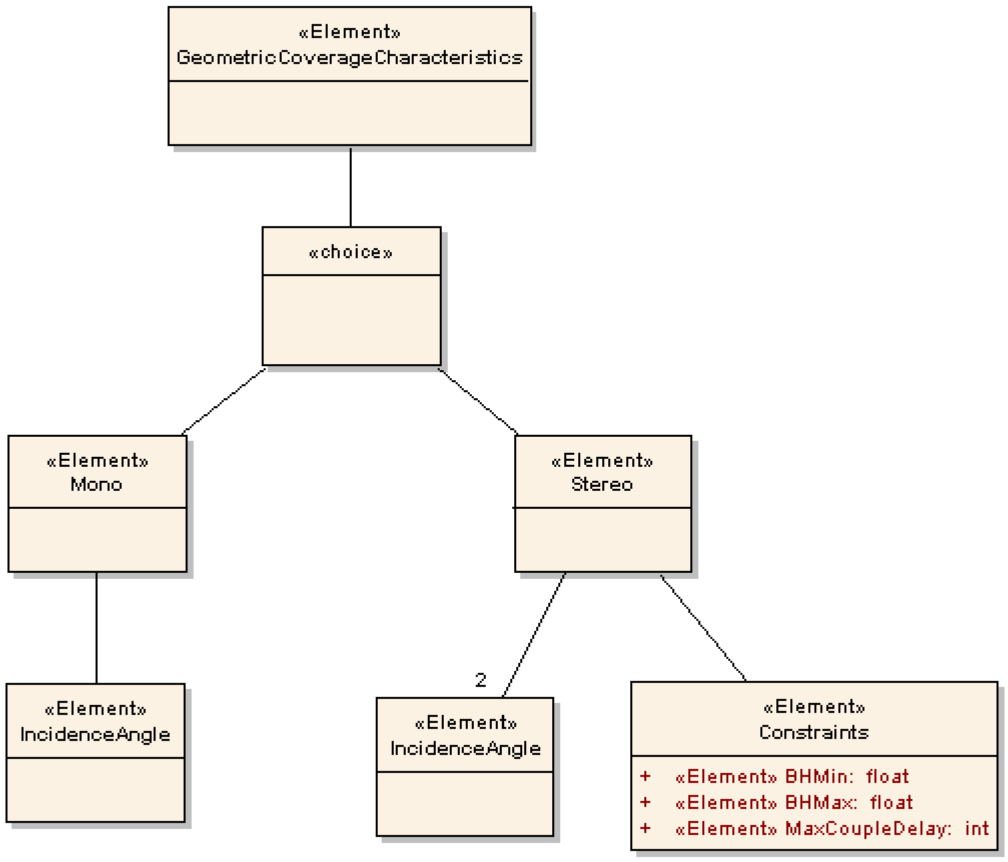
8.2.2.2 OHR mission (optical)


«Element»
AcquisitionParameters OHR
+ «Element» Resolution: float
+ «Element» AcquisitionMode: enum string
+ «Element» ProgrammingRequestMode: enum string
«Element»
SurveyPeriods

Tasking Parameter Name	Description	Source	Mission
Resolution	Sensor resolution Type: floating point with allowed values	[NR11]	OHR
AcquisitionMode	Type: Enumerated String Possible values are: • MULTISPECTRAL • PANCHROMATIC • BUNDLE	[NR11]	OHR

Tasking Parameter Name	Description	Source	Mission
ProgrammingRequestMode	Programming type Type: Enumerated String Possible values are: • SCENE: unique attitude acquisition, for tasks specified by time intervals / orbit segments • MONOPASS COVERAGE: coverage acquired through a unique pass; many attitude acquisition necessary • MULTIPASS COVERAGE: coverage may be acquired through several passes	[NR11]	OHR
SurveyPeriods	cf § 8.2.6		

8.2.3 GeometricCoverageCharacteristics



Tasking Parameter Name	Description	Source	Mission
Mono	Mono acquisition	[OR4]	All
IncidenceAngle	cf. § 8.2.6		
Stereo	Stereo acquisition	[OR4]	ohr
IncidenceAngle1	cf. § 8.2.6		
IncidenceAngle2	cf. § 8.2.6		
Constraints			
BHMin	Minimum base over Height ratio accepted between a stereo pair. Type: float		

Tasking Parameter Name	Description	Source	Mission
BHMax	Maximum base over Height ratio accepted between a stereo pair. Type: float		
MaxCoupleDelay	Maximum interval of days between two stereo acquisitions. Type: integer		

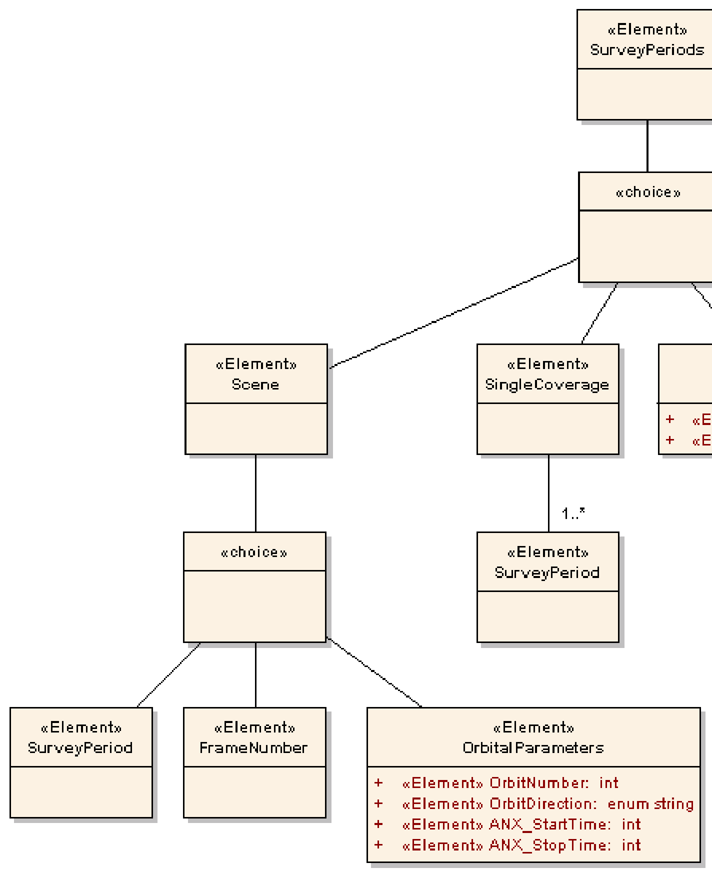
8.2.4 RegionOfInterest

Tasking Parameter Name	Description	Source	Mission
Region of Interest	Type: GeometryDefinition (polygon, circle, TBD)	[OR4]	all

8.2.5 ValidationParameters

Tasking Parameter Name	Description	Source	Mission
cloudCoverPercentage	Maximum allowed cloud coverage Type: floating point Unit: percentage	[OR4]	ohr
snowCoverPercentage	Maximum allowed snow coverage Type: floating point Unit: percentage	[OR4]	ohr
hazeAccepted	Haze presence accepted Type: boolean		ohr
sandWindAccepted	Sand wind presence accepted Type: boolean		ohr

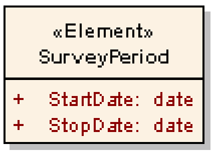
8.2.6 SurveyPeriods type


«Element» SurveyPeriods
«choice»
«Element» Scene
«Element» SingleCoverage
«Element» SeveralCoverages
+ «Element» Occurence: int
+ «Element» FrozzenDays: int
«Element» TemporalSeries
+ «Element» Periodicity: int
«choice»
1..*
«Element» SurveyPeriod
«Element» SurveyPeriod
«Element» SurveyPeriod
«Element» SurveyPeriod
«Element» FrameNumber
«Element» OrbitalParameters
+ «Element» OrbitNumber: int
+ «Element» OrbitDirection: enum string
+ «Element» ANX_StartTime: int
+ «Element» ANX_StopTime: int

Tasking Parameter Name	Description	Source	Mission
Scene	Acquisition of a single scene	[OR4]	all
SurveyPeriod	Start date and end date of acquisition Type: PeriodType (cf § 8.2.7)		all
FrameNumber			all
OrbitalParameters			all
OrbitNumber	Orbit number Type: integer	[NR11]	all
OrbitDirection	Orbit direction Type: Enumerated String Possible values: ASCENDING, DESCENDING	[NR11]	all
ANX_StartTime	Time in millisecond of the acquisition start with respect the ascending node crossing. ANX_StartTime & ANX_StopTime are alternative to startDate & completionDate. Type: integer Unit: milliseconds	[NR11]	all
ANX_StopTime	Time in millisecond of the acquisition end with respect the ascending node crossing ANX_StartTime & ANX_StopTime are alternative to startDate & completionDate. Type: integer Unit: milliseconds	[NR11]	all

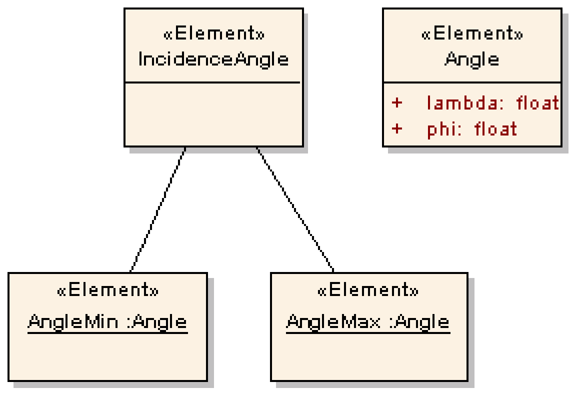
Tasking Parameter Name	Description	Source	Mission
SingleCoverage	One (mono date survey) or several (multi date survey) observation periods are requested.	[OR4]	all
SurveyPeriod	Start date and end date of acquisition Type: PeriodType (cf § 8.2.7)		all
SeveralCoverages	Only one observation periods is requested, but inside this period, the area should be covered more than once, eventually respecting a frozen period between two acquisitions.		all
SurveyPeriod	Start date and end date of acquisition Type: PeriodType (cf § 8.2.7)		all
Occurrence	Number of repetitions of the reference observation period required (including the reference observation as first observation). The reference observation period is defined by startDate & completionDate. Type: integer		all
FrozzenDays	Interval between two consecutive observations. Type: integer Unit: day Precision: day		all
TemporalSeries			
SurveyPeriod	Start date and end date of acquisition Type: PeriodType (cf § 8.2.7)		
Periodicity	Acquisition periodicity. Type: unsigned integer Unit: days		

8.2.7 Period type



Tasking Parameter Name	Description	Source	Mission
StartDate	Start acquisition date. Type: date	[OR4]	all
EndDate	End acquisition date. Type: date	[OR4]	all

8.2.8 Incidence angle type



Tasking Parameter Name	Description	Source	Mission
AngleMin lambda phi	Minimum incidence angle. Type: floating point Unit: degrees	[OR4]	all
AngleMax lambda phi	Maximum incidence angle Type: floating point Unit: degrees	[OR4]	all

8.3 Input parameters

The Parameter Element is used to provide the value for a specific parameter. The encoding follows the description that is part of the definition element of a ParameterDescriptor Element. The Parameter Element is therefore rather simple in its definition. It just has to provide the mandatory parameterID attribute to link the values to the specific parameter. The values itself are replacing the any-Element.

The following figures illustrate the Parameter element structure.

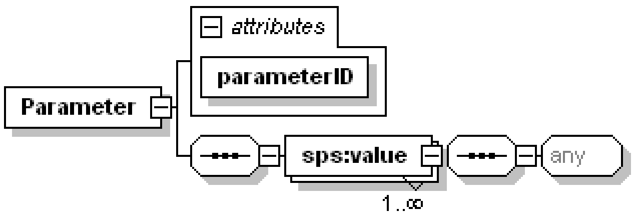


Figure 8-3 - InputParameter element diagram

Example:

```
<Parameter parameterID="maxCloudCoverage">
  <value>
    <swe:Count>10</swe:Count>
  </value>
</Parameter>
```

8.4 sensorID

The sensorID is an identifier to a set of tasking parameters. A sensorID may represent a sensor (example: SPOT 5), one instrument of one satellite (example: SPOT 5 HRS) or a constellation of satellites (example SPOT+FORMOSAT). The sensorID may also represent a combination of other sensorIDs (an example can be found in § 20.1.1.1).

Depending on what the sensor ID represents, the list of input parameters may be different. For instance if a sensorID represents a constellation of satellites (example: SPOT2+SPOT4+SPOT5), one input parameter may be the index of the satellite (example: 5). But in case of the sensorID represents one instrument of one satellite, there will be no need for such input parameter.

8.5 DeliveryInformationType

This type has been derived by the DeliveryInformationType defined in the Ordering Service ([NR12]).

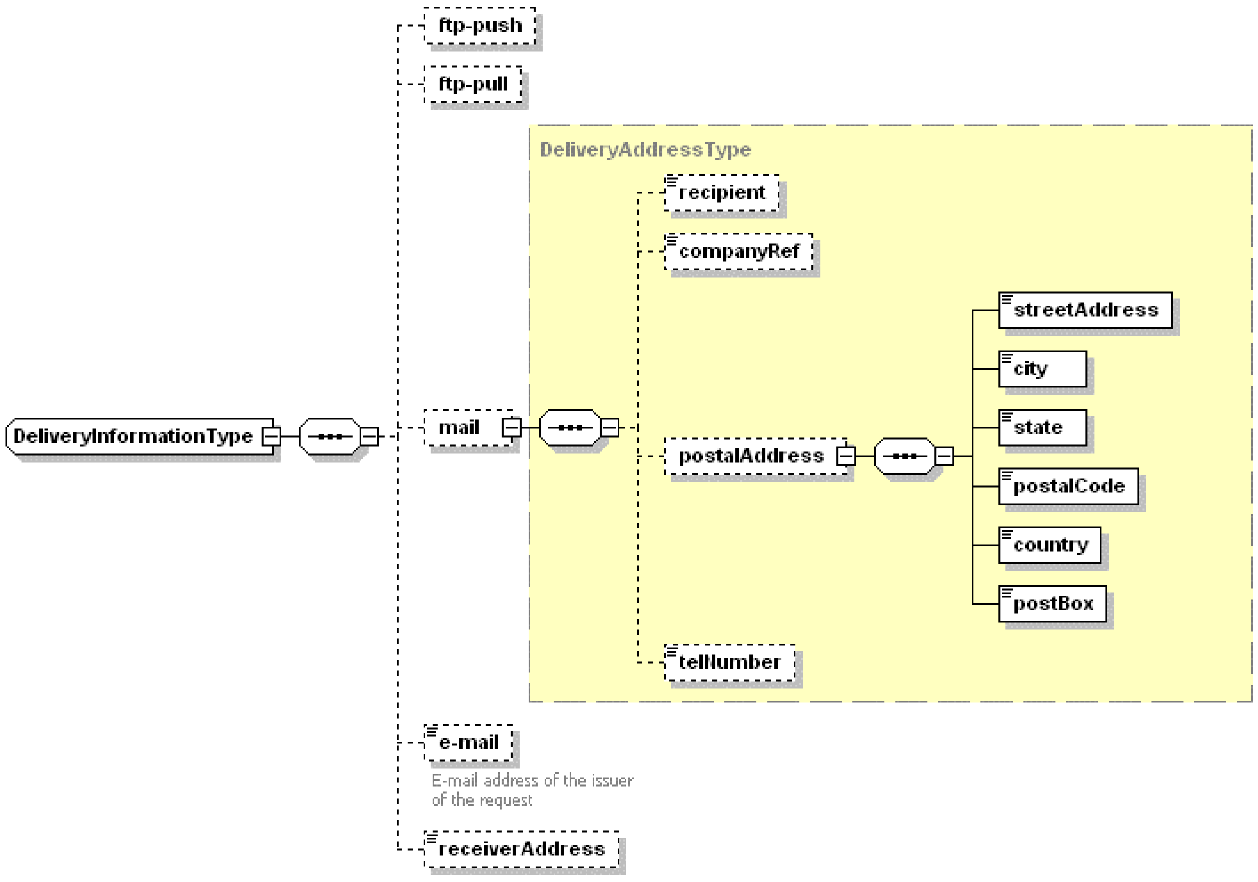


Figure 8-4: DeliveryInformationType diagram.

Tag Name	Tag Description
ftp-push	FTP URL: address of a user-owned FTP server to which a product can be posted containing also directory, username, password information. **Type:** Not empty string (max 255 chars) **Syntax :** **ftp:**//'ftpUserName':'ftpPassword'@'ftpAddress'/'ftpDirectory' **Example: ftp:**//muis_intecs:intecs@ftp.intecs.it/MUIS
ftp-pull	FTP URL: address of a provider-owned FTP server from which user can fetch products containing also directory, username, password information. The value is set by the provider, therefore the element has to be set to <blank> in the SubmitRequest **Type:** string (max 255 chars) **Syntax:** **ftp:**//'ftpUserName':'ftpPassword'@'ftpAddress'/'ftpDirectory' **Example: ftp:**//userOder:userpwd@ftp.esa.int/XI/EN1
mail	Mail element.
Recipient	Identification of the receiving person. **Type:** Not empty string (max 40 chars)
companyRef	Identification of the receiving entity.
postalAddress	Postal Address of the user.
streetAddress	Street Adress element. **Type:** String
city	City element. **Type:** String

Tag Name	Tag Description
state	State element. **Type:** String
postalCode	Postal Code element. **Type:** String
country	Country element. **Type:** String
postalBox	Postal Box element. **Type:** String
telNumber	Telephone number of the receiving person. **Type:** Not empty string (max 18 chars) matching the following regular expression: "\+?[0-9\(\)-\s]+" (An optional "+" sign followed by a series of (at least one) digit, "(", ")", "-" and blank chars)
e-mail	E-mail address of the user. Type: String
receiverAddress	DDS address Type: String

Table 8-1: DeliveryInformationType description.

9 External interfaces

This clause describes the externally visible behaviour of the system, including the interfaces implemented by its components and the supported protocol bindings. It defines the request and response message structures as part of the operation --signatures, primarily the differences to those of the OpenGIS® Sensor Planning Service Implementation Specification [NR13].

9.1 Imported protocol bindings (relationship with SPS implementation specification [NR13])

This profile inherits the HTTP/SOAP protocol binding from the Sensor Planning Service Implementation Specification [NR13].

The following table reports the mapping of SPS operation on the Programming Service operations.

SPS operations	Programming Service operations
GetCapabilities	GetCapabilities
	DescribeSensor
DescribeGetFeasibility	DescribeGetFeasibility
GetFeasibility	GetFeasibility
	EstimateSensorWorkload
ReserveTasking	
DescribeSubmit	DescribeSubmit
Submit	Submit
GetStatus	GetStatus
Update	Update
Cancel	Cancel
DescribeResultAccess	DescribeResultAccess

Table 9-1: Mapping of Programming Service to SPS operations.

All operations except GetCapabilities must support the embedding of requests and responses in SOAP messages. Only SOAP messaging (via HTTP/POST) with document/literal style has to be used. Messages must conform to SOAP 1.2 (http://www.w3.org/TR/SOAP/). The message payload will be in the body of the SOAP envelope:

9.2 Operations interface

The following table shows the main characteristics of the operations. Each operation is fully described further in the document.

operation name	Request encoding	Protocol	Mandatory	Sync/ Async
GetCapabilities	KVP	HTTP/GET	X	S
DescribeSensor	XML	SOAP messaging via HTTP/POST	X	S
DescribeGetFeasibility	XML	SOAP messaging via HTTP/POST		S
GetFeasibility	XML	SOAP messaging via HTTP/POST		A
EstimateSensorWorkload	XML	SOAP messaging via HTTP/POST		S
DescribeSubmit	XML	SOAP messaging via HTTP/POST	X	S
Submit	XML	SOAP messaging via HTTP/POST	X	A
GetStatus	XML	SOAP messaging via HTTP/POST		S
Update	XML	SOAP messaging via HTTP/POST		S
Cancel	XML	SOAP messaging via HTTP/POST		S
DescribeResultAccess	XML	SOAP messaging via HTTP/POST	X	S

9.3 Shared aspects

9.3.1 Protocol

In the HMA context, all requests and responses are encapsulated into SOAP messages, except for the GetCapabilities operation.

9.3.2 notificationTarget versus WS-Addressing (asynchronous operations)

The SPS specification imposes a mandatory **notificationTarget** parameter used to identify the Web Notification Service that will send notifications to the client in case of asynchronous operations.

In the HMA context, SOAP with WS-Addressing protocol has been chosen in order to communicate asynchronously, for the following reasons:

- WS-Addressing is a W3C recommendation (http://www.w3.org/2002/ws/addr)
- WS-Addressing defines a standard for incorporating message addressing information into web services messages
- WS-Addressing defines standard ways to route a message over multiple transports or direct a response to a third party. For example, a client application might send a request over JMS and ask to receive the response through e-mail or SMS (http://dev2dev.bea.com/pub/a/2005/01/ws_addressing_intro.html)
- WS-Security can be used with WS-Addressing
- WS-Addressing is very easy to implement

- In the message, there is a "physical" separation between the notification stuff (in the header of the message) and the OGC request (in the body).
- Any request of any service can be asynchronous without any modification of the request (just add a replyAddress in the header, that's it !)

Therefore there is no need for a WNS and the notificationTarget parameter must be optional.

The header of a SOAP message using WS-Addressing contains a ***messageID*** and a ***replyAddress*** :

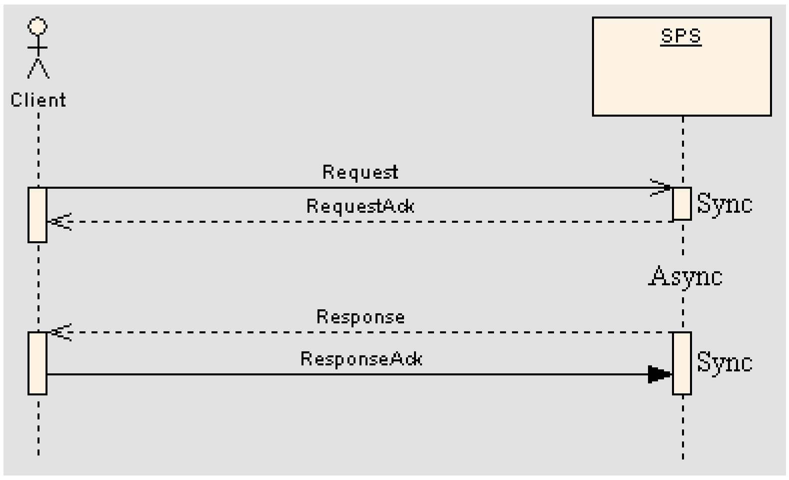
```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header xmlns:wsa="http://www.w3.org/2004/12/addressing">
   <wsa:MessageID>LZH789</wsa:MessageID>
    <wsa:ReplyTo>
        <wsa:Address>http://ws.spotimage.com/client_sps</wsa:Address>
    </wsa:ReplyTo>
  </soapenv:Header>
  <soapenv:Body>OWS request</soapenv:Body >
</soapenv:Envelope>
```

The response is sent asynchronously to the address specified in the ***Address*** element of the header. The header of the SOAP response message contains a ***relatesTo*** element for the ***messageID***, allowing the client to link the request and the asynchronous response :

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header xmlns:wsa="http://www.w3.org/2004/12/addressing">
   <wsa:RelatesTo>LZH789</wsa:RelatesTo>
  </soapenv:Header>
  <soapenv:Body>OWS response</soapenv:Body >
</soapenv:Envelope>
```

9.3.3 Acknowlegments

Asynchronous operations are defined by a request and an asynchronous response. Acknowledgements will have to be sent upon request synchronously. This allows the sender of the message to make sure the message is successfully received:



The RequestAck and ResponseAck acknowledgments may contain more elements, but they contain at least a status element described below:

9.3.3.1 Request acknowledgment status

Request acknowledgment status schema:

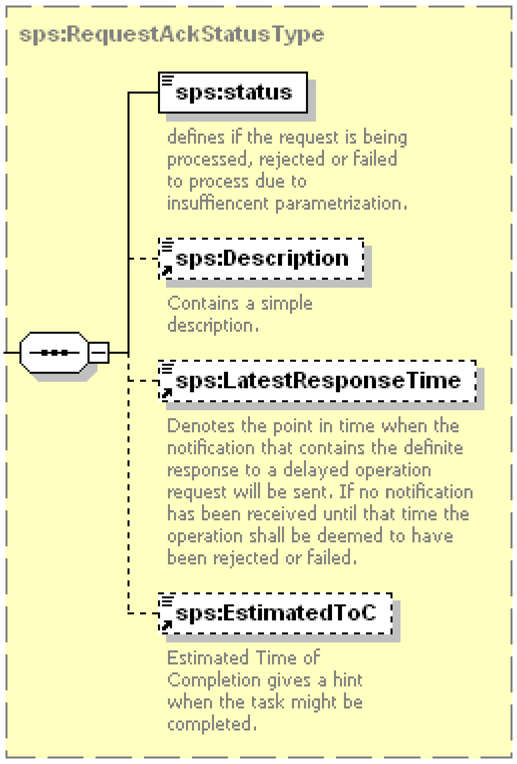

sps:RequestAckStatusType
sps:status
defines if the request is being processed, rejected or failed to process due to insuffiencent parametrization.
sps:Description
Contains a simple description.
sps:LatestResponseTime
Denotes the point in time when the notification that contains the definite response to a delayed operation request will be sent. If no notification has been received until that time the operation shall be deemed to have been rejected or failed.
sps:EstimatedToC
Estimated Time of Completion gives a hint when the task might be completed.

Figure 9-1 - Request acknowledgment status schema

Table 9-2: Request acknowledgement status

Name	Definition	Data type and values	Multiplicity and use
status	Status of the request	String enumerates: “confirmed” “rejected” “incomplete request” “pending” “rejected, alternatives available”	one (mandatory)
Description	Text description of the response	StringOrRefType	one (optional)
Latest Response Time	GetFeasibility response will be sent until LatestResponseTime at latest. In case that no response is received, the operation shall be evaluated as non-feasible.	gml:TimeInstantType	one (optional)
estimatedToC	Defines estimated time of completion	gml:TimeInstant	one (optional)

9.3.3.2 Response acknowledgment status

Response acknowledgment status schema:

Figure 9-2 - Response acknowledgement status schema

Table 9-3: Response acknowledgement status

Name	Definition	Data type and values	Multiplicity and use
Response AckStatus	Indicates that the Response message was successfully received.	String Possible value is : Ok (in case of the Response message was not successfully received, an exception is thrown)	one (mandatory)

9.3.4 Progress reports

Operations Submit and GetStatus return information about the status of a request. This information is defined as a ProgressReport specified as follows:

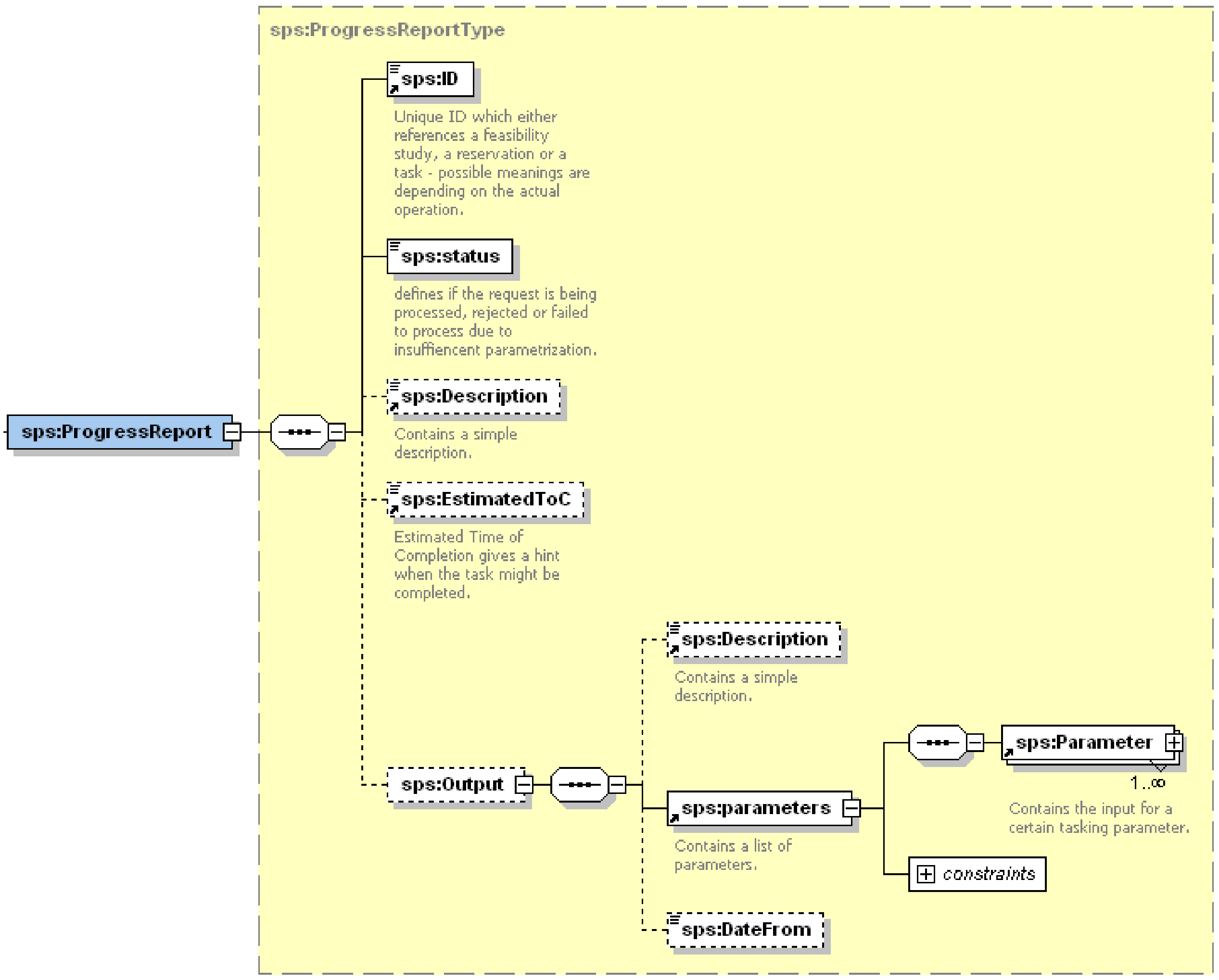

sps:ProgressReportType
sps:ProgressReport
sps:ID
Unique ID which either references a feasibility study, a reservation or a task - possible meanings are depending on the actual operation.
sps:status
defines if the request is being processed, rejected or failed to process due to insuffiencent parametrization.
sps:Description
Contains a simple description.
sps:EstimatedToC
Estimated Time of Completion gives a hint when the task might be completed.
sps:Output
sps:Description
Contains a simple description.
sps:parameters
Contains a list of parameters.
sps:Parameter
1..∞
Contains the input for a certain tasking parameter.
constraints
sps:DateFrom

Figure 9-3 - Progress report schema

The following table describes the elements of the Progress report schema:

Table 9-4: Progress report

Name	Definition	Data type and values	Multiplicity and use
ID	Identifier for this task, needed for subsequent update requests (taskID).	token	one (mandatory)
status	Identifier of the status of the request	String enumerates: “confirmed” “rejected” “incomplete request” “pending” “rejected, alternatives available”	one (mandatory)
Description	Additional metadata	String or reference to external source	one (optional)
Latest ResponseTime	Submit response will be sent until LatestResponseTime at latest. In case that no response is received, the operation shall be evaluated as rejected.	gml:TimeInstantType	one (mandatory)
estimatedToC	Defines estimated time of completion	gml:TimeInstant	one (optional)
Output			
Description			one (optional)
parameters	List of acquired scenes since the date defined in the DateFrom input parameter. These parameters are described in the **OutputParameters** element of the DescribeSubmit response.		
DateFrom	Date from which the result is given.	Date	one (optional)

10 GetCapabilities operation (mandatory, synchronous)

10.1 Introduction

The mandatory GetCapabilities operation allows clients to retrieve service metadata from a server. The response to a GetCapabilities request shall be an XML document containing service metadata about the server, including specific information about a SPS. This clause specifies the XML document that a SPS server must return to describe its capabilities.

10.2 EO profile specific

- The description of the sensors has been removed from the capabilities (the description is returned by the DescribeSensor operation);
- The content section of the capabilities contains the supported communication protocols. For SPS EO profile, the communication protocol is SOAP with WS-Addressing.
- PhenomenonOfferingList is removed

10.3 Capabilities schema

The following diagram shows the content of the response sent by the SPS to a GetCapabilities request:

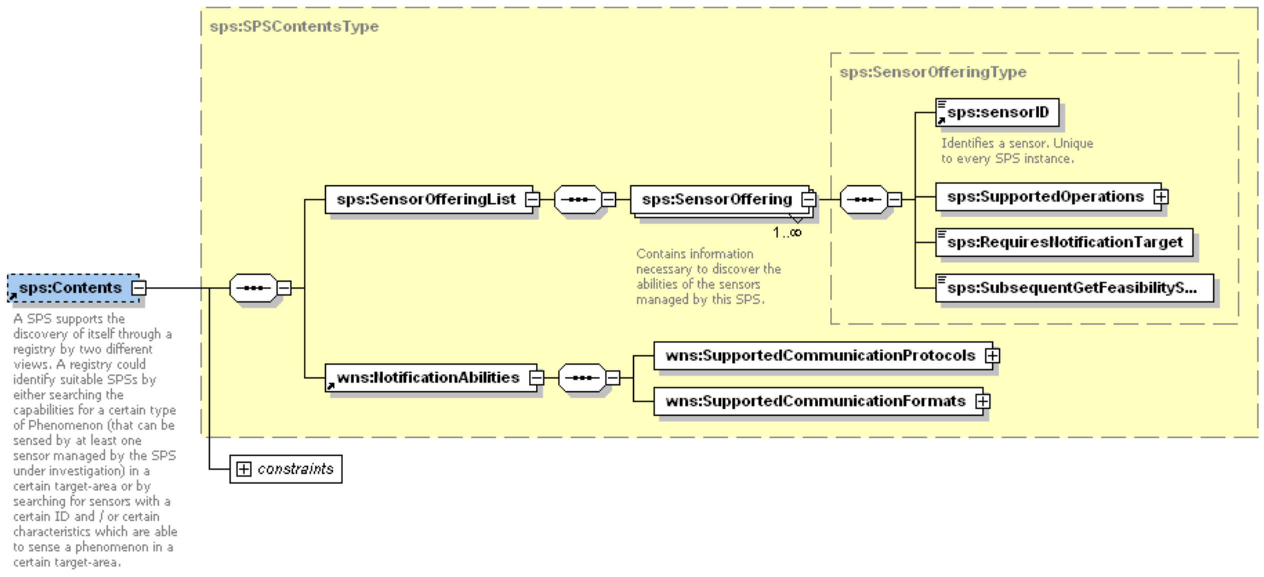

sps:SPSContentsType
sps:SensorOfferingType
sps:sensorID
Identifies a sensor. Unique to every SPS instance.
sps:SensorOfferingList
sps:SensorOffering
1..∞
Contains information necessary to discover the abilities of the sensors managed by this SPS.
sps:SupportedOperations
sps:RequiresNotificationTarget
sps:SubsequentGetFeasibilityS...
sps:Contents
A SPS supports the discovery of itself through a registry by two different views. A registry could identify suitable SPSs by either searching the capabilities for a certain type of Phenomenon (that can be sensed by at least one sensor managed by the SPS under investigation) in a certain target-area or by searching for sensors with a certain ID and / or certain characteristics which are able to sense a phenomenon in a certain target-area.
wns:NotificationAbilities
wns:SupportedCommunicationProtocols
wns:SupportedCommunicationFormats
constraints

Figure 9-4 - Capabilities schema

10.4 GetCapabilities Operation request

The GetCapabilities operation request shall be as specified in Subclauses 7.2 and 7.3 of [OGC 05-008]. The value of the "service" parameter shall be "SPS". The allowed set of service metadata (or Capabilities) XML document section names and meanings shall be as specified in Tables 3 and 7 of [OGC 05-008].

The “Multiplicity and use” column in Table 1 of [OGC 05-008] specifies the optionality of each listed parameter in the GetCapabilities operation request. The following table specifies the implementation of those parameters by Programming Service clients and servers.

Name	Multiplicity	Client implementation	Server implementation
service	One (mandatory)	Each parameter shall be implemented by all clients, using specified value	Each parameter shall be implemented by all servers, checking that each parameter is received with specified value
request	One (mandatory)		
AcceptVersions	Zero or one (optional)	Should be implemented by all software clients, using specified values	Shall be implemented by all servers, checking if parameter is received with specified value(s)
Sections	Zero or one (optional)	Each parameter may be implemented by each client If parameter not provided, shall expect default response If parameter provided, shall allow default or specified response	Each parameter may be implemented by each server If parameter not implemented or not received, shall provide default response If parameter implemented and received, shall provide specified response
updateSequence	Zero or one (optional)		
AcceptFormats	Zero or one (optional)		

Table 9-5: Implementation of parameters in GetCapabilities operation request

All SPS servers shall implement HTTP GET transfer of the GetCapabilities operation request, using KVP encoding. Servers may also implement HTTP POST transfer of the GetCapabilities operation request, using XML encoding only.

10.5 GetCapabilities Operation response

10.5.1 Normal response

The service metadata document shall contain the SPS sections specified in the following table. Depending on the values in the Sections parameter of the GetCapabilities operation request, any combination of these sections can be requested and shall be returned when requested.

Table 9-6— Section name values and contents

Section name	Contents
ServiceIdentification	Metadata about this specific server. The schema of this section shall be the same as for all OWSs, as specified in Subclause 7.4.3 and owsServiceIdentification.xsd of [OGC 05-008].
ServiceProvider	Metadata about the organization operating this server. The schema of this section shall be the same for all OWSs, as specified in Subclause 7.4.4 and owsServiceProvider.xsd of [OGC 05-008].
OperationsMetadata	Metadata about the operations specified by this service and implemented by this server, including the URLs for operation requests. The basic contents and organization of this section shall be the same as for all OWSs, as specified in Subclause 7.4.5 and owsOperationsMetadata.xsd of [OGC 05-008].
Contents	Metadata about the data served by this server. For the SPS, this section shall contain information about the sensors that can be tasked and the phenomena that can be measured by these sensors, as specified in Subclause 0 below.

	NotificationAbilities.
SensorOfferingList	Information necessary to discover the abilities of the sensors managed by the SPS.
sensorID	Cf. § 8.4
Supported Operations	List of optional operations implemented by the service
Requires Notification Target	TBC
Subsequent GetFeasibility Supported	TBC
Notification Abilities	HMA context: value=WS-Addressing

In addition to these sections, each service metadata document shall include the mandatory "version" and optional updateSequence parameters specified in Table 6 in Subclause 7.4.1 of [OGC 05-008].

10.5.2 OperationsMetadata section standard contents

For the SPS, the OperationsMetadata section shall be the same as for all OGC Web Services, as specified in Subclause 7.4.5 and owsOperationsMetadata.xsd of [OGC 05-008].

Attribute name	Attribute value	Meaning of attribute value
Operation. name	GetCapabilities	The GetCapabilities operation is implemented by this server.
	DescribeGetFeasibility	The DescribeGetFeasibility operation is implemented by the server
	DescribeSubmit	The DescribeSubmit operation is implemented by the server
	Submit	The Submit operation is implemented by this server.
	DescribeResultAccess	The DescribeResultAccess operation is implemented by this server.

Table 9-7: Mandatory Programming Service operations.

Attribute name	Attribute value	Meaning of attribute value
	DescribeSensor	The DescribeSensor operation is implemented by the server
	GetFeasibility	The GetFeasibility operation is implemented by this server.
	GetStatus	The GetStatus operation is implemented by this server.
	Update	The Update operation is implemented by this server
	Cancel	The Cancel operation is implemented by this server.

Table 9-8: Optional Programming Service operations.

10.6 Exceptions

When a SPS server encounters an error while performing a GetCapabilities operation, it shall return an exception report message as specified in Clause 8 of [OGC 05-008].

10.7 Example of GetCapabilities response

```
<Capabilities version="1.0.30"
    xmlns="http://www.opengis.net/sps"
    xmlns:ows="http://www.opengeospatial.net/ows"
    xmlns:wns="http://www.opengeospatial.net/wns"
    xmlns:gml="http://www.opengis.net/gml"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation=" http://www.opengis.net/sps/1.0.30 ./spsAll.xsd">
  <ows:ServiceIdentification>
    <ows:Title>Spot Image SPS EO profile Prototype</ows:Title>
    <ows:ServiceType>SPS</ows:ServiceType>
    <ows:ServiceTypeVersion>1.0.30</ows:ServiceTypeVersion>
    <ows:Fees>none</ows:Fees>
    <ows:AccessConstraints>none</ows:AccessConstraints>
  </ows:ServiceIdentification>
  <ows:ServiceProvider>
    <ows:ProviderName>Spot Image</ows:ProviderName>
    <ows:ProviderSite>http://www.spotimage.com</ows:ProviderSite>
    <ows:ServiceContact>
      <ows:IndividualName>Philippe Merigot</ows:IndividualName>
    </ows:ServiceContact>
  </ows:ServiceProvider>
  <ows:OperationsMetadata>
    <ows:Operation name="GetCapabilities">
      <ows:DCP>
        <ows:HTTP>
          <ows:Get          xlink:href="http://ws.spotimage.com/sisa_ws2/sps"
xlink:type="simple"/>
        </ows:HTTP>
      </ows:DCP>
    </ows:Operation>
    <ows:Operation name="DescribeSensor">
      <ows:DCP>
        <ows:HTTP>
          <ows:Post         xlink:href="http://ws.spotimage.com/sisa_ws2/sps"
xlink:type="simple"/>
        </ows:HTTP>
      </ows:DCP>
    </ows:Operation>
    <ows:Operation name="DescribeGetFeasibility">
      <ows:DCP>
        <ows:HTTP>
          <ows:Post         xlink:href="http://ws.spotimage.com/sisa_ws2/sps"
xlink:type="simple"/>
        </ows:HTTP>
      </ows:DCP>
    </ows:Operation>
    <ows:Operation name="DescribeSubmit">
      <ows:DCP>
        <ows:HTTP>
          <ows:Post         xlink:href="http://ws.spotimage.com/sisa_ws2/sps"
xlink:type="simple"/>
        </ows:HTTP>
      </ows:DCP>
    </ows:Operation>
    <ows:Operation name="GetFeasibility">
      <ows:DCP>
        <ows:HTTP>
          <ows:Post          xlink:href="http://ws.spotimage.com/sisa_ws2/sps"
xlink:type="simple"/>
        </ows:HTTP>
      </ows:DCP>
    </ows:Operation>
    <ows:Operation name="Submit">
      <ows:DCP>
        <ows:HTTP>
          <ows:Post          xlink:href="http://ws.spotimage.com/sisa_ws2/sps"
xlink:type="simple"/>
        </ows:HTTP>
      </ows:DCP>
    </ows:Operation>
    <ows:Operation name="DescribeResultAccess">
      <ows:DCP>
        <ows:HTTP>
          <ows:Post          xlink:href="http://ws.spotimage.com/sisa_ws2/sps"
xlink:type="simple"/>
        </ows:HTTP>
      </ows:DCP>
    </ows:Operation>
    <ows:Operation name="GetStatus">
      <ows:DCP>
        <ows:HTTP>
          <ows:Post          xlink:href="http://ws.spotimage.com/sisa_ws2/sps"
xlink:type="simple"/>
        </ows:HTTP>
      </ows:DCP>
    </ows:Operation>
    <ows:Operation name="Update">
      <ows:DCP>
        <ows:HTTP>
          <ows:Post          xlink:href="http://ws.spotimage.com/sisa_ws2/sps"
xlink:type="simple"/>
        </ows:HTTP>
      </ows:DCP>
    </ows:Operation>
    <ows:Operation name="Cancel">
      <ows:DCP>
        <ows:HTTP>
          <ows:Post          xlink:href="http://ws.spotimage.com/sisa_ws2/sps"
xlink:type="simple"/>
        </ows:HTTP>
      </ows:DCP>
    </ows:Operation>
  </ows:OperationsMetadata>
  <Contents>
    <SensorOfferingList>
      <SensorOffering>
       <sensorID>urn:ogc:object:feature:Sensor:SpotImage:spot5</sensorID>
       <SupportedOperations    DescribeSubmit="true"    GetFeasibility="true"
Submit="true"      Update="false"      GetStatus="true"       Cancel="false"
DescribeResultAccess="true"/>
       <RequiresNotificationTarget>false</RequiresNotificationTarget>
   <SubsequentGetFeasibilitySupported>true</SubsequentGetFeasibilitySupported>
      </SensorOffering>
    </SensorOfferingList>
    <wns:NotificationAbilities>
```

```
      <wns:SupportedCommunicationProtocols>
        <wns:XMPP>false</wns:XMPP>
        <wns:SMS>false</wns:SMS>
        <wns:Phone>false</wns:Phone>
        <wns:Fax>false</wns:Fax>
        <wns:Email>false</wns:Email>
        <wns:WSAddressing>true</wns:WSAddressing>
        <wns:WNS>false</wns:WNS>
      </wns:SupportedCommunicationProtocols>
      <wns:SupportedCommunicationFormats>
        <wns:NotificationFormat>basic</wns:NotificationFormat>
      </wns:SupportedCommunicationFormats>
    </wns:NotificationAbilities>
  </Contents>
</Capabilities>
```

11 DescribeSensor operation (mandatory, synchronous)

11.1 Introduction

This operation returns SensorML documents containing the description of the sensors provided by the SPS.

Depending on the user's choice, this operation may return either a full or a brief description of the sensors.

The full description contains all the details of the sensors (geometry, agility, swath, lists and definitions of observables supported by the sensor, etc.). It should be used only in specific cases (simulation of the sensors for instance).
The brief description contains a limited number of information describing the nature of the sensor (for example : optical satellite, 10 meter resolution, 4 bands). This allows the user to ensure that a specific sensor fits its needs. Due to the limited number of information, the SensorML document returned by the server is light and can be exchanged on the network and parsed by the client quickly.

An example of full and brief description can be found in annex B.

11.2 EO profile specific

The operation DescribeSensor is specific to the EO profile.

11.3 DescribeSensor operation request

Table 11-1- Parameters of DescribeSensor Request

Name [a]	Definition	Data type and values	Multiplicity and use
SensorId	The sensorId parameter specifies the sensor for which the description is to be returned.	token	One (mandatory)
descriptionType	Type of description that the server should return.	String Possible values: – brief – full	One (mandatory)
service	Service type identifier	Character String Fixed value : SPS	One (mandatory)
version	Specification version for operation	Character String type, not empty Value is specified by each Implementation Specification and Schemas version	One (mandatory)
a The name capitalization rules being used here are specified in Subclause 11.6.2 of [OGC 05-008].			

11.3.1 DescribeSensor request XML encoding

This example illustrates requesting information about a specific sensor instance.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <DescribeSensor                service="SPS"                version="1.0.30"
xmlns="http://www.opengis.net/sps">
      <sensorID>urn:ogc:object:feature:Sensor:SpotImage:spot.xs</sensorID>
      <descriptionType>brief</ descriptionType >
    </DescribeSensor>
  </soapenv:Body>
</soapenv:Envelope>
```

11.4 DescribeSensor operation response

Table 11-2- Parts of DescribeSensor operation response

Name	Definition	Data type and values	Multiplicity and use
sensorID	Identifies the sensor that shall be tasked	token	one, mandatory
description	Provides SensorML description	complex	one to many, mandatory

An example DescribeSensor response can be found in annex B.

11.5 Exceptions

When a SPS server encounters an error while performing a DescribeSensor operation, it shall return an exception report message as specified in clause 8 of [OGC 05-008].

11.6 DescribeSensor response example

A example of DescribeSensor operation response for SPS is provided on Annex B.

12 EstimateSensorWorkload operation (optional, synchronous)

The EstimateSensorWorkload provides information about the estimated workload of the requested sensor. This information is under the responsibility of each mission and must be accessible on demand by the SPS server. The SPS server has to maintain up to date information of such workload by harvesting the information to the mission on regular basis (weekly, or monthly)
The minimal information provided by the mission is as follows:

- Temporal domain: begin date, end date,
- Temporal resolution: number of days (i.e. how many days between two consecutive values in a mesh),
- Spatial domain: lat min, lat max, long min, long max (somewhat similar to "Area of service"),
- Spatial resolution: size of the meshes, expressed in degrees, minutes or in km

Further on, the following information has to be provided for each mesh:

- mesh location: lat-long coordinates of the mesh centre,
- mesh date or range of dates
- workload value: percentage of the resource already booked on this mesh at this date

12.1 EO profile specific

The operation EstimateSensorWorkload is specific to the EO profile.

12.2 EstimateSensorWorkload operation request

Table 12-1- Attributes of EstimateSensorWorkload Request

Name [a]	Definition	Data type and values	Multiplicity and use
outputFormat	The outputFormat attribute specifies the desired output format of the EstimateSensorWorkload operation. For the EO Profile the format will be SensorML	Character String type	Mandatory
SensorId	The sensorId parameter specifies the sensor for which the description is to be returned.	token	One (mandatory)
service	Service type identifier	Character String Fixed value : SPS	One (mandatory)
version	Specification version for operation	Character String type, not empty Value is specified by each Implementation Specification and Schemas version	One (mandatory)
a The name capitalization rules being used here are specified in Subclause 11.6.2 of [OGC 05-008].			

12.2.1 EstimateSensorWorkload request XML encoding

This example illustrates requesting information about a specific sensor instance.

```
<EstimateSensorWorkload                    version="1.0.30"                service="SPS"
outputFormat="text/xml;subtype=sensorML/1.0.0"
xmlns="http://www.opengeospatial.net/sps">
  <SensorId>urn:ogc:object:feature:Sensor:SpotImage:spot.XS</SensorId>
</EstimateSensorWorkload>
```

12.3 EstimateSensorWorkload operation response

Table 12-2- Parts of EstimateSensorWorkload operation response

Name	Definition	Data type and values	Multiplicity and use
sensorID	Identifies the sensor that shall be tasked	token	one, mandatory
description	Provides EstimateSensorWorkload description	complex	one to many, mandatory

12.4 Exceptions

When a SPS server encounters an error while performing a EstimateSensorWorkload operation, it shall return an exception report message as specified in clause 8 of [OGC 05-008].

12.5 EstimateSensorWorkload response example

A EstimateSensorWorkload operation response for SPS can look like this encoded in XML:

```
<EstimateSensorWorkloadResponse              xmlns="http://www.opengis.net/sps"
xmlns:xlink="http://www.w3.org/1999/xlink">
   <SensorID>
   urn:ogc:object:feature:Sensor:SpotImage:spot10M
   </sensorID>
   <sml:SensorML                    xmlns:sml="http://www.opengis.net/sensorML"
xmlns:swe="http://www.opengis.net/swe"    xmlns:gml="http://www.opengis.net/gml"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xlink="http://www.w3.org/1999/xlink"
xsi:schemaLocation="http://www.opengis.net/sensorML
http://vast.uah.edu/schemas/sensorML/1.0.30/base/sensorML.xsd" version="1.0">
          <SensorID>
             urn:ogc:object:feature:Sensor:SpotImage:spot10M
          </SensorID>
                      <!-------------------------------------->
                      <!--EstimateSensorWorkload Inputs-->
                      <!-------------------------------------->
</EstimateSensorWorkloadResponse>
```

13 DescribeGetFeasibility operation (optional, synchronous)

13.1 Introduction

The DescribeGetFeasibility operation allows SPS clients to request the information necessary to prepare a programming request (targeted at the sensors that are supported by the SPS and that are selected by the client). The server will return information about all parameters that have to be set by the client in order to perform a GetFeasibility operation and eventually the description of parameters returned in the GetFeasibility response. The only additional parameter "SensorID" defines the specific sensor(s) that shall be described by the server. This allows servers to façade multiple sensors that require parameterization and return all information to the client using one call only.

Because GetFeasibility operation is optional, DescribeGetFeasibility is also optional.

13.2 EO profile specific

- The ParameterDescriptor has been modified (see § 8.1).
- DescribeGetFeasibility response: InputParameters and OutputParameters share the same (EO profile specific) type **ParametersType**
- DescribeGetFeasibility response: **InputParameters** is optional and should not be used if the parameters are the same as the Submit input parameters.

13.3 DescribeGetFeasibility operation schemas

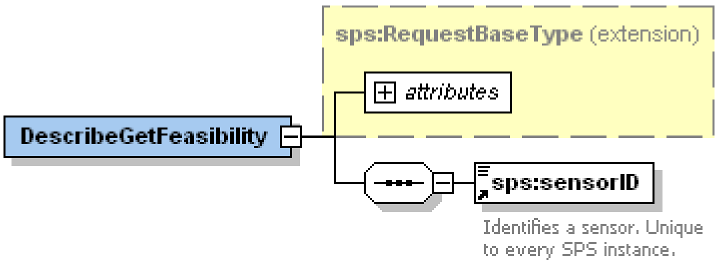

sps:RequestBaseType (extension)
attributes
DescribeGetFeasibility
sps:sensorID
Identifies a sensor. Unique to every SPS instance.

Figure 13-1: - DescribeGetFeasibility request schema

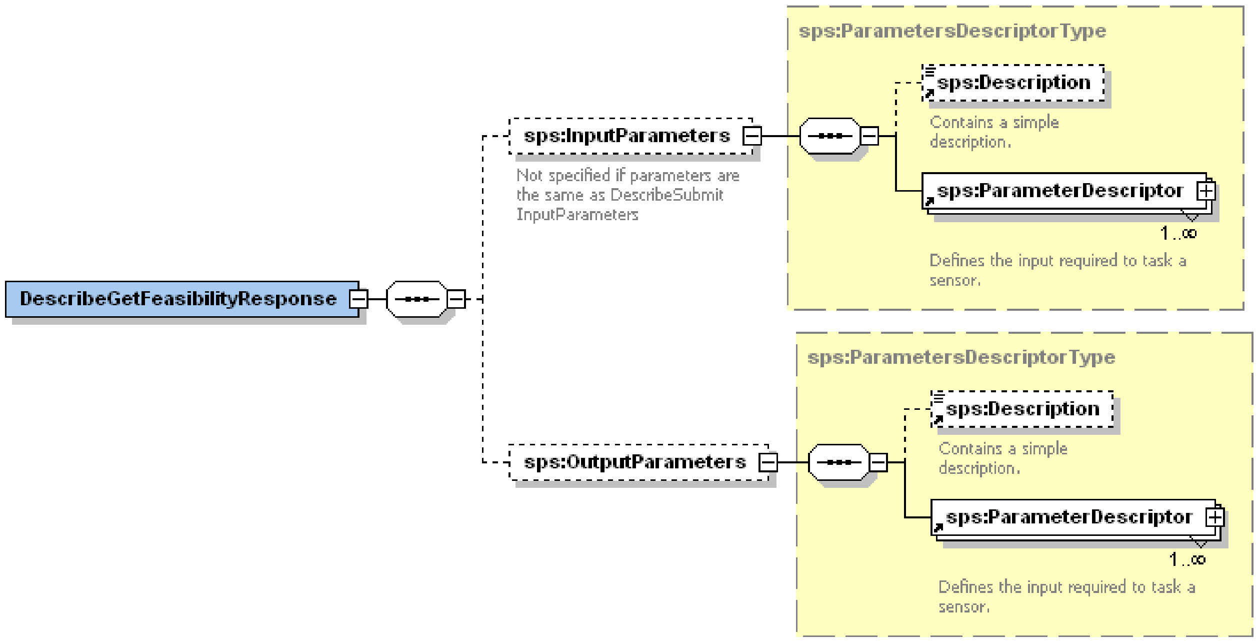

DescribeGetFeasibilityResponse
sps:InputParameters
Not specified if parameters are the same as DescribeSubmit InputParameters
sps:OutputParameters
sps:ParametersDescriptorType
sps:Description
Contains a simple description.
sps:ParameterDescriptor
1..∞
Defines the input required to task a sensor.
sps:ParametersDescriptorType
sps:Description
Contains a simple description.
sps:ParameterDescriptor
1..∞
Defines the input required to task a sensor.

Figure 13-2: - DescribeGetFeasibility response schema

13.4 DescribeGetFeasibility operation request

Table 13-1: - Parameters in DescribeGetFeasibility operation request

Name [a]	**Definition**	**Data type and values**	**Multiplicity and use**
service	Service type identifier	Character String type, not empty fixed: Value is OWS type abbreviation: SPS	One (mandatory)
version	Specification version for operation	Character String type, not empty Value is specified by each Implementation Specification and Schemas version	One (mandatory)
sensorID	Defines sensor to be described	token	One (mandatory)
a The name capitalization rules being used here are specified in Subclause 11.6.2 of [OGC 05-008].			

EXAMPLE

```
<DescribeGetFeasibility service="SPS" version="1.0.30">
  <sensorID>urn:ogc:object:feature:Sensor:SpotImage:spot10M</sensorID>
</ DescribeGetFeasibility>
```

13.5 DescribeGetFeasibility operation response

If the client provides a valid sensorID (cf. capabilities), the SPS server will respond with a DescribeGetFeasibilityResponse.

Table 13-2: Parts of DescribeGetFeasibility operation response

Name	Definition	Data type and values	Multiplicity and use
InputParameters	Provides the parameters of a GetFeasibility request	Complex type Contains a description and a list of ParameterDescriptor	one, optional
OutputParameters	Describe the parameters returned in the response. EO profile: list of acquired scenes.	Complex type Contains a description and a list of ParameterDescriptor	one, optional

13.6 DescribeGetFeasibility exceptions

When a SPS server encounters an error while performing a DescribeGetFeasibility operation, it shall return an exception report message as specified in Subclause 7.4 of [OGC 05-008].

14 DescribeSubmit operation (mandatory, synchronous)

14.1 Introduction

The DescribeSubmit operation request allows SPS clients to request the information necessary to prepare a programming request targeted at the sensors that are supported by the SPS and that are selected by the client. The server will return information about all parameters that have to be set by the client in order to perform a Submit operation.

Because Submit operation is mandatory, DescribeSubmit is also mandatory.

14.2 EO profile specific

- DescribeSubmit response: a new element **InputParameters** is added at the root for homogeneity with the DescribeGetFeasibility response. It contains a list of **ParameterDescriptor** elements.

- DescribeSubmit response: a new element **OutputParameters** is added at the root. It allows the client to get the description of the output parameters returned by a Submit operation (element *Output* in the response).

14.3 DescribeSubmit operation schemas

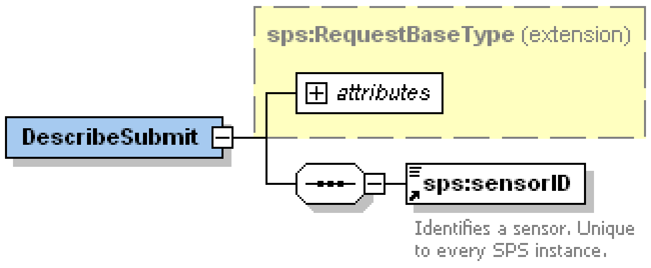

sps:RequestBaseType (extension)
attributes
DescribeSubmit
sps:sensorID
Identifies a sensor. Unique to every SPS instance.

Figure 14-1 - DescribeSubmit request schema

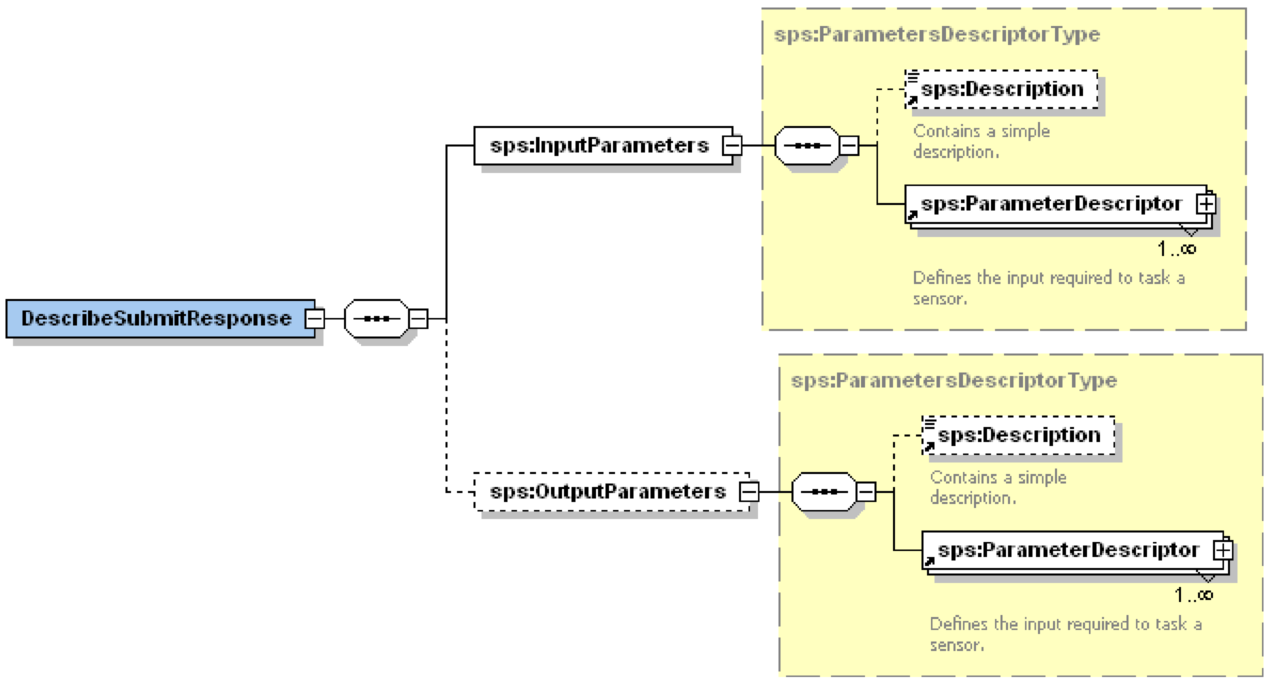

DescribeSubmitResponse
sps:InputParameters
sps:OutputParameters
sps:ParametersDescriptorType
sps:Description
Contains a simple description.
sps:ParameterDescriptor
1..∞
Defines the input required to task a sensor.
sps:ParametersDescriptorType
sps:Description
Contains a simple description.
sps:ParameterDescriptor
1..∞
Defines the input required to task a sensor.

Figure 14-2 - DescribeSubmit response schema

14.4 DescribeSubmit operation request

Table 14-1: - Parameters in DescribeSubmit operation request

Name [a]	Definition	Data type and values	Multiplicity and use
service	Service type identifier	Character String type, not empty fixed: Value is OWS type abbreviation: SPS	One (mandatory)
version	Specification version for operation	Character String type, not empty Value is specified by each Implementation Specification and Schemas version	One (mandatory)
sensorID	Defines sensor to be described	token	One (mandatory)
a The name capitalization rules being used here are specified in Subclause 11.6.2 of [OGC 05-008].			

14.5 DescribeSubmit operation response

Table 14-2: Parts of DescribeSubmit operation response

Name	Definition	Data type and values	Multiplicity and use
InputParameters	Provides the parameters of a Submit request	Complex type Contains a description and a list of ParameterDescriptor	one, mandatory
OutputParameters	Describe the parameters returned in the Output element of the Submit and the GetStatus responses. EO profile: list of acquired scenes.	Complex type Contains a description and a list of ParameterDescriptor	one, optional

An example of DescribeSubmit response can be found in annex B of this document.

14.6 DescribeGetFeasibility exceptions

When a SPS server encounters an error while performing a DescribeSubmit operation, it shall return an exception report message as specified in Subclause 7.4 of [OGC 05-008].

15 GetFeasibility and Submit operations

15.1 EO profile specific use cases

Both GetFeasibility and Submit operations are described in more detail in subsequent clauses. The following figures show the EO profile specific use cases:

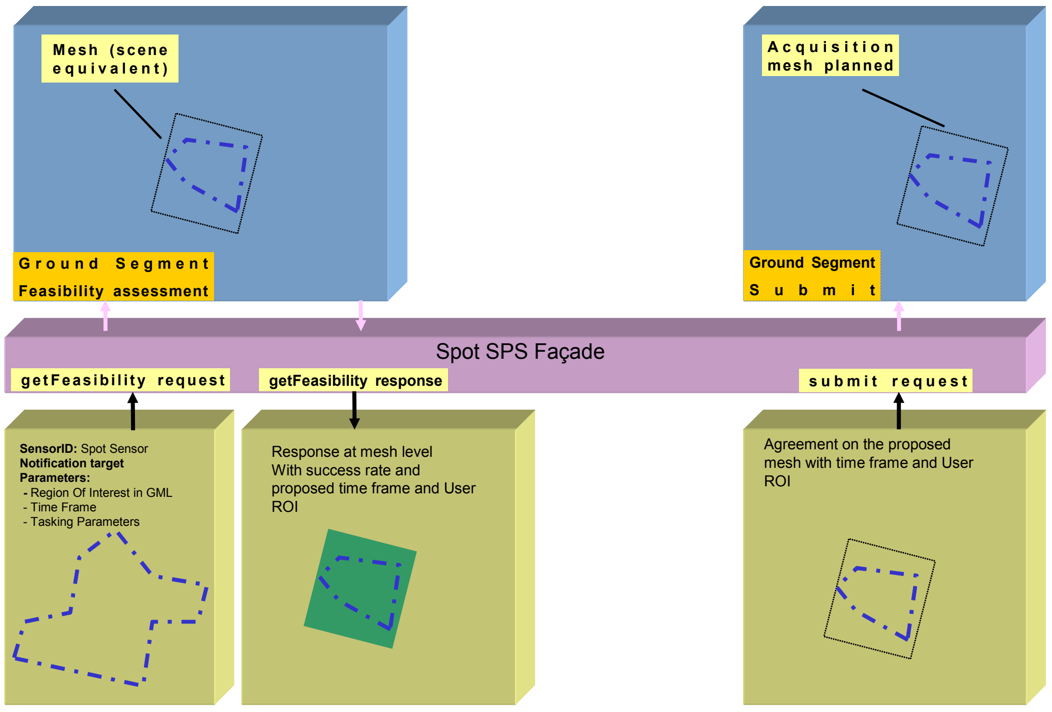


Figure 15-1: - SPS for Earth Observation Sensor: getFeasibility/Submit operations: scene use case

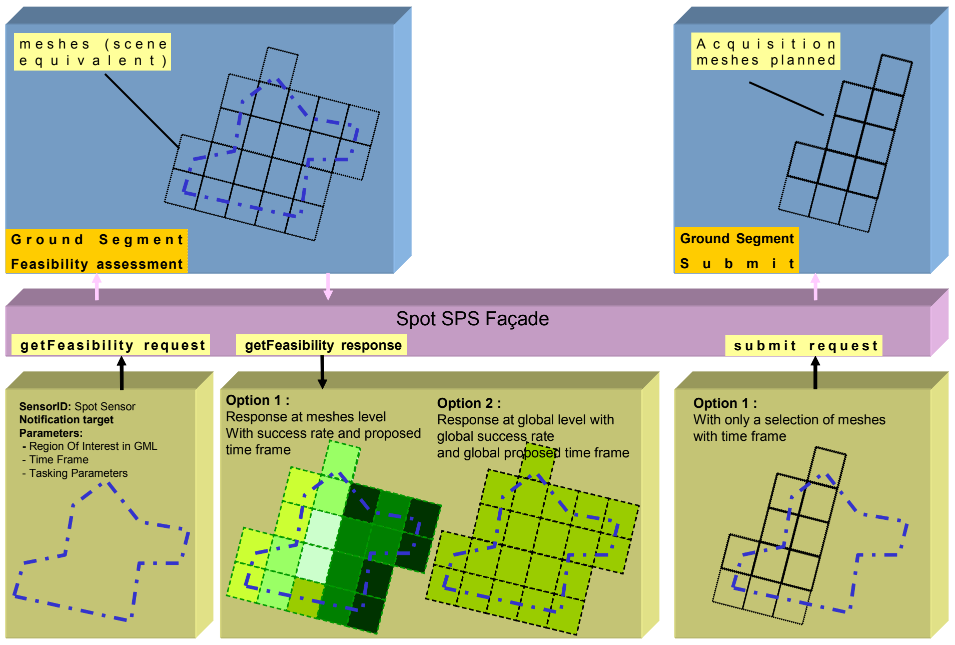


Figure 15-2: - SPS for Earth Observation Sensor: getFeasibility/Submit operations: coverage use case

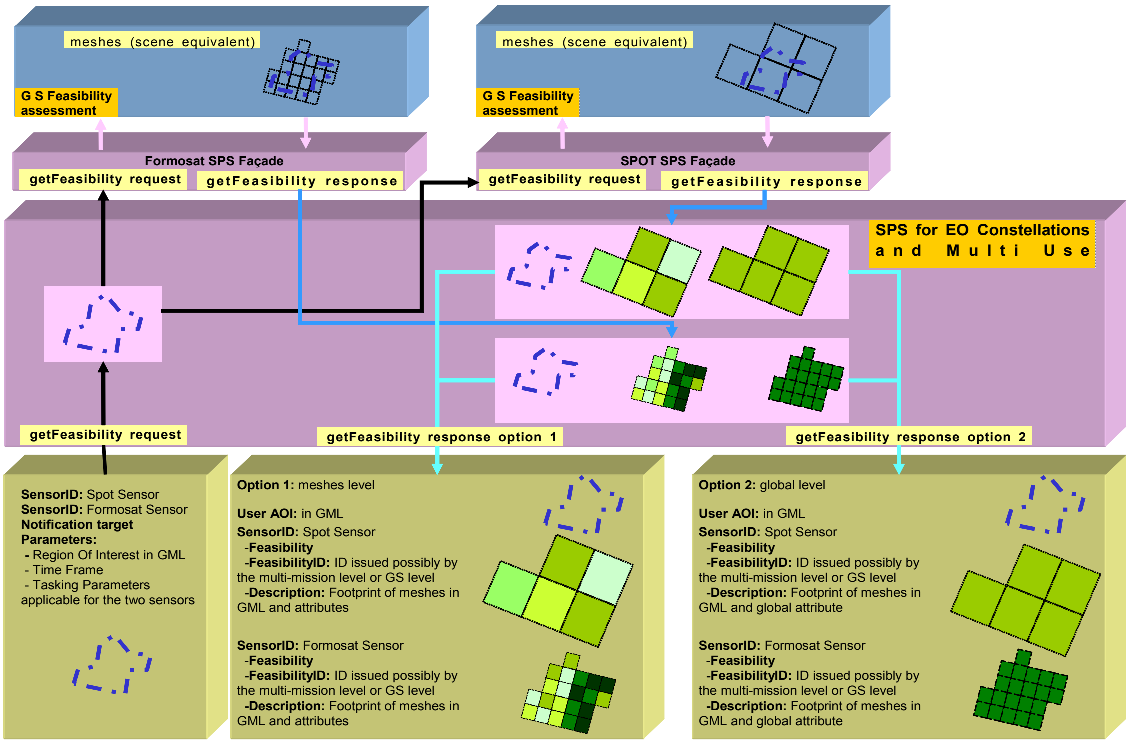

meshes (scene equivalent)
G S Feasibility assessment
meshes (scene equivalent)
G S Feasibility assessment
Formosat SPS Façade
getFeasibility request
getFeasibility response
SPOT SPS Façade
getFeasibility request
getFeasibility response
SPS for EO Constellations and Multi Use
getFeasibility request
getFeasibility response option 1
getFeasibility response option 2
SensorID: Spot Sensor
SensorID: Formosat Sensor
Notification target
Parameters:
- Region Of Interest in GML
- Time Frame
- Tasking Parameters applicable for the two sensors
Option 1: meshes level
User AOI: in GML
SensorID: Spot Sensor
-Feasibility
-FeasibilityID: ID issued possibly by the multi-mission level or GS level
-Description: Footprint of meshes in GML and attributes
SensorID: Formosat Sensor
-Feasibility
-FeasibilityID: ID issued possibly by the multi-mission level or GS level
-Description: Footprint of meshes in GML and attributes
Option 2: global level
User AOI: in GML
SensorID: Spot Sensor
-Feasibility
-FeasibilityID: ID issued possibly by the multi-mission level or GS level
-Description: Footprint of meshes in GML and global attribute
SensorID: Formosat Sensor
-Feasibility
-FeasibilityID: ID issued possibly by the multi-mission level or GS level
-Description: Footprint of meshes in GML and global attribute

Figure 15-3: - SPS for multi sensors: getFeasibility operation detailed coverage use case with invocation of the GS

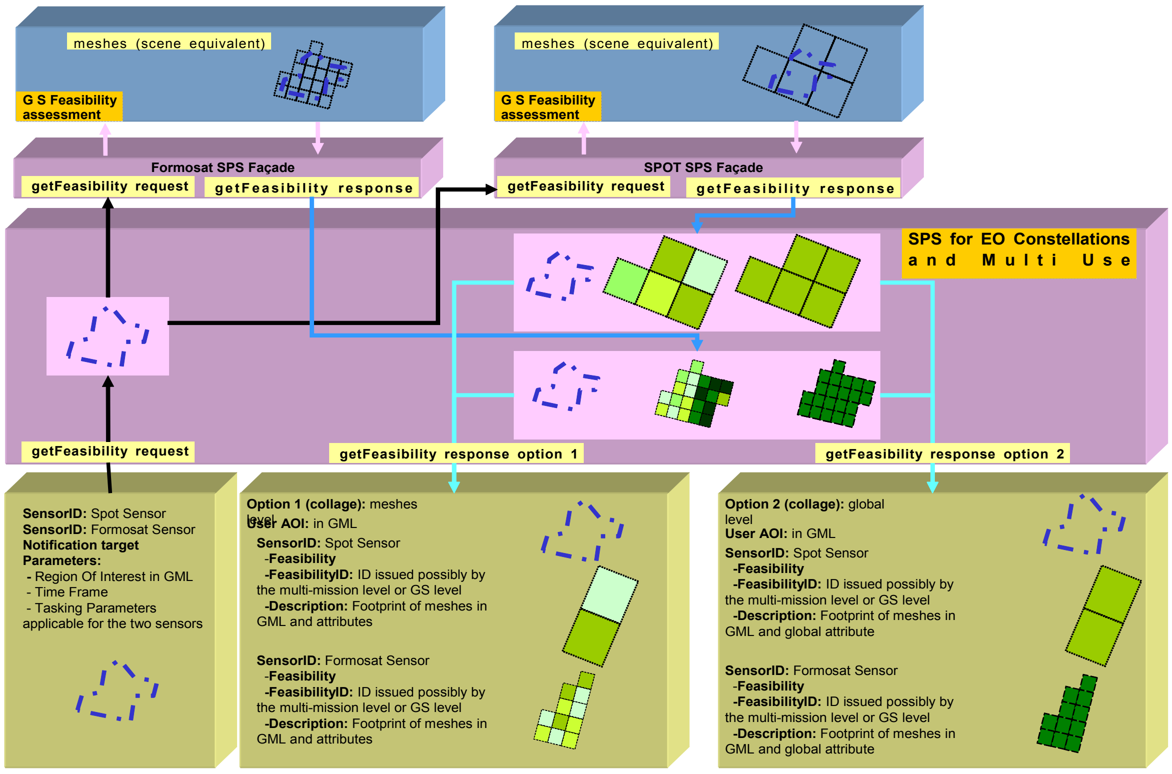

meshes (scene equivalent)
G S Feasibility assessment
meshes (scene equivalent)
G S Feasibility assessment
Formosat SPS Façade
getFeasibility request
getFeasibility response
SPOT SPS Façade
getFeasibility request
getFeasibility response
SPS for EO Constellations and Multi Use
getFeasibility request
getFeasibility response option 1
getFeasibility response option 2
SensorID: Spot Sensor
SensorID: Formosat Sensor
Notification target
Parameters:
- Region Of Interest in GML
- Time Frame
- Tasking Parameters applicable for the two sensors
Option 1 (collage): meshes level
User AOI: in GML
SensorID: Spot Sensor
-Feasibility
-FeasibilityID: ID issued possibly by the multi-mission level or GS level
-Description: Footprint of meshes in GML and attributes
SensorID: Formosat Sensor
-Feasibility
-FeasibilityID: ID issued possibly by the multi-mission level or GS level
-Description: Footprint of meshes in GML and attributes
Option 2 (collage): global level
User AOI: in GML
SensorID: Spot Sensor
-Feasibility
-FeasibilityID: ID issued possibly by the multi-mission level or GS level
-Description: Footprint of meshes in GML and global attribute
SensorID: Formosat Sensor
-Feasibility
-FeasibilityID: ID issued possibly by the multi-mission level or GS level
-Description: Footprint of meshes in GML and global attribute

Figure 15-4: - SPS for multi sensors: getFeasibility operation detailed coverage use case with invocation of the GS

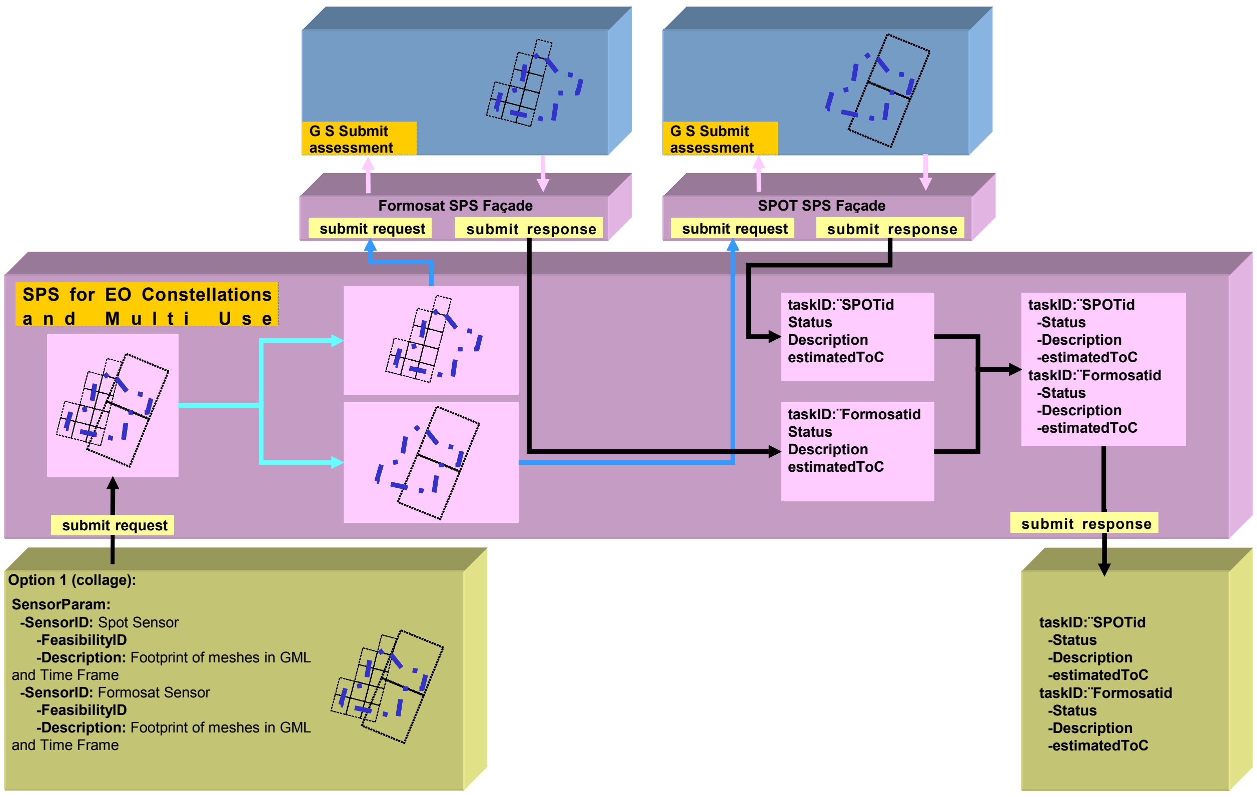

G S Submit assessment
G S Submit assessment
Formosat SPS Façade
submit request
submit response
SPOT SPS Façade
submit request
submit response
SPS for EO Constellations and Multi Use
taskID:¨SPOTid Status Description estimatedToC
taskID:¨Formosatid Status Description estimatedToC
taskID:¨SPOTid -Status -Description -estimatedToC taskID:¨Formosatid -Status -Description -estimatedToC
submit request
submit response
Option 1 (collage):
SensorParam:
-SensorID: Spot Sensor
-FeasibilityID
-Description: Footprint of meshes in GML and Time Frame
-SensorID: Formosat Sensor
-FeasibilityID
-Description: Footprint of meshes in GML and Time Frame
taskID:¨SPOTid -Status -Description -estimatedToC taskID:¨Formosatid -Status -Description -estimatedToC

Figure 15-5: - SPS for multi sensors getFeasibility operation: coverage use case

15.2 GetFeasibility operation (optional, asynchronous)

15.2.1 Introduction

The GetFeasibility operation allows SPS clients to obtain information about the feasibility of a programming request.

GetFeasibility uses an asynchronous communication model, as illustrated in the figure below. When the SPS server receives the request, an acknowledgment is sent back by the SPS and the connection is closed. When the SPS has finished its feasibility study later on, it will open a new connection (i.e. asynchronously), to send the response back to the client, which itself acknowledges the reception of the response message.

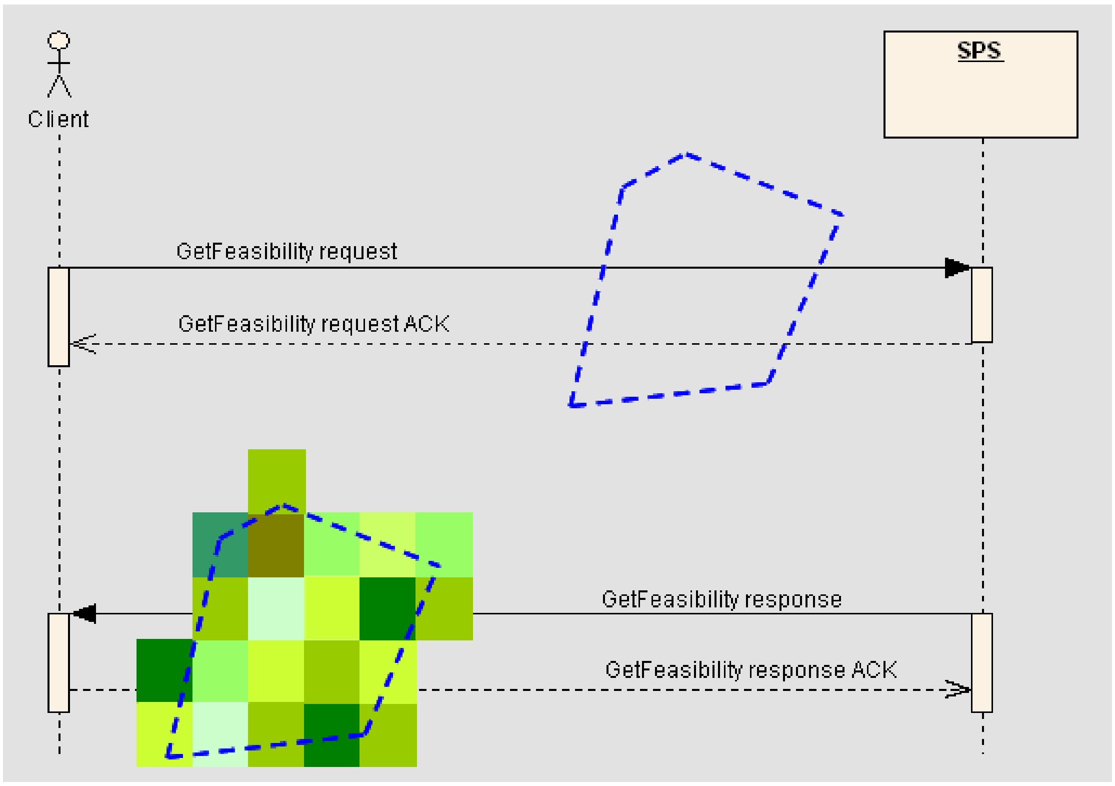


Figure 15-6 - GetFeasibility asynchronous communication model

The overall communication process includes four messages (described further in the document):

1. the request: *GetFeasibility* (§ 15.2.3)
2. the acknowledgment of the request: *GetFeasibilityAck* (§ 15.2.4)
3. the response (callback): *GetFeasibilityResponse* (§ 15.2.5)
4. the acknowledgment of the response: *GetFeasibilityResponseAck* (§ 15.2.6)

15.2.2 EO profile specific

- Acknowledgments
- The GetFeasibility request contains a new element named **feasibilityLevel**
- WS-Addressing is used instead of the **wns:NotificationTarget** parameter (see § 9.3.2).
- The GetFeasibility response contains the list of scenes which may be acquired to cover the regionOfInterest specified by the client in the Submit operation:

15.2.3 GetFeasibility request

Schema of the GetFeasibility request:

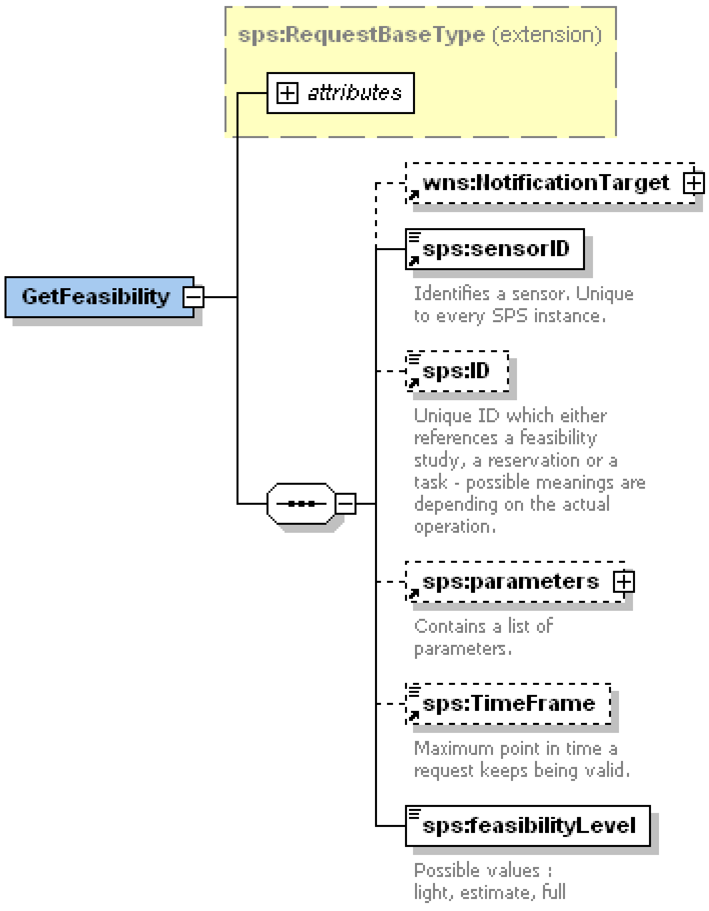

sps:RequestBaseType (extension)
attributes
wns:NotificationTarget
sps:sensorID
Identifies a sensor. Unique to every SPS instance.
GetFeasibility
sps:ID
Unique ID which either references a feasibility study, a reservation or a task - possible meanings are depending on the actual operation.
sps:parameters
Contains a list of parameters.
sps:TimeFrame
Maximum point in time a request keeps being valid.
sps:feasibilityLevel
Possible values : light, estimate, full

Figure 15-7: - GetFeasibility request

The following table describes the elements of the GetFeasibility request schema:

Table 15-1: Parameters in GetFeasibility operation request

Name [a]	**Definition**	**Data type and values**	**Multiplicity and use**
service	Service type identifier	Character String type, not empty Value is "SPS"	One (mandatory)
version	Specification version for operation	Character String type, not empty Value is same as version of this document	One (mandatory)
notificationTarget	Defines the WNS that has to be used to notify the client about the request results	complex type	one (optional)
sensorID	Identifies the sensor that shall be tasked	token	one (mandatory)
ID	Identifier for the feasibility study	token	one (optional) use for re-iterate feasibility request with new parameters after an alternative proposal from the GS
parameters	Input parameter values. Encoding should match description in **InputParameters** DescribeGetFeasibility response (or DescribeSubmit response if DescribeGetFeasibility is not implemented)	complex See § 8	one (mandatory)
feasibilityLevel	Defines the feasibility level	Allowed values: • **light** : parameters check • **estimate** : climatology (if applicable) + estimated workload • **full** : climatology (if applicable) + workload	one (mandatory)
timeFrame	From ([NR13]): Maximum point in time the request keeps valid.		one (optional)
a The name capitalization rules being used here are specified in Subclause 11.6.2 of [OGC 05-008].			

15.2.4 GetFeasibility request acknowledgment

Schema of the GetFeasibility request acknowledgment:

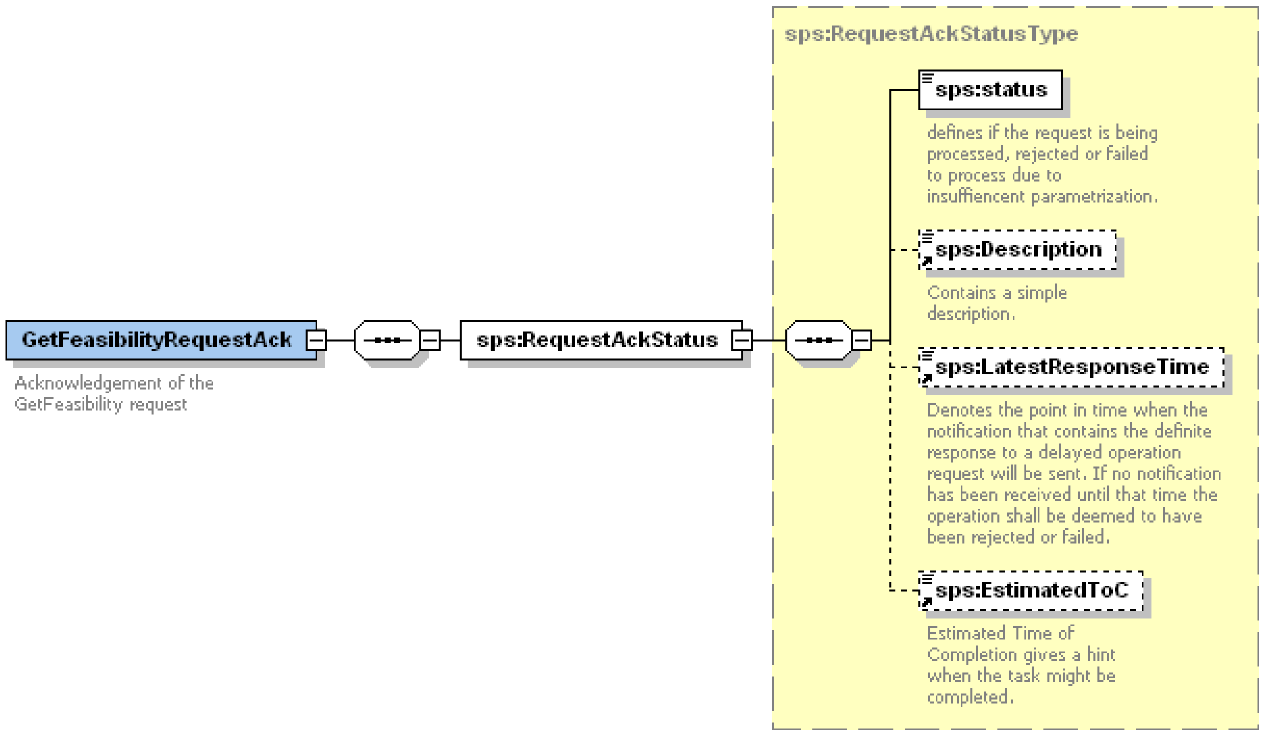

sps:RequestAckStatusType
GetFeasibilityRequestAck
Acknowledgement of the GetFeasibility request
sps:RequestAckStatus
sps:status
defines if the request is being processed, rejected or failed to process due to insuffiencent parametrization.
sps:Description
Contains a simple description.
sps:LatestResponseTime
Denotes the point in time when the notification that contains the definite response to a delayed operation request will be sent. If no notification has been received until that time the operation shall be deemed to have been rejected or failed.
sps:EstimatedToC
Estimated Time of Completion gives a hint when the task might be completed.

Figure 15-8 - GetFeasibility request acknowledgment schema

The RequestAckStatus element is described in § *9.3.3.1 Request acknowledgment status*.

.

15.2.5 GetFeasibility response

Schema of the GetFeasibility response:

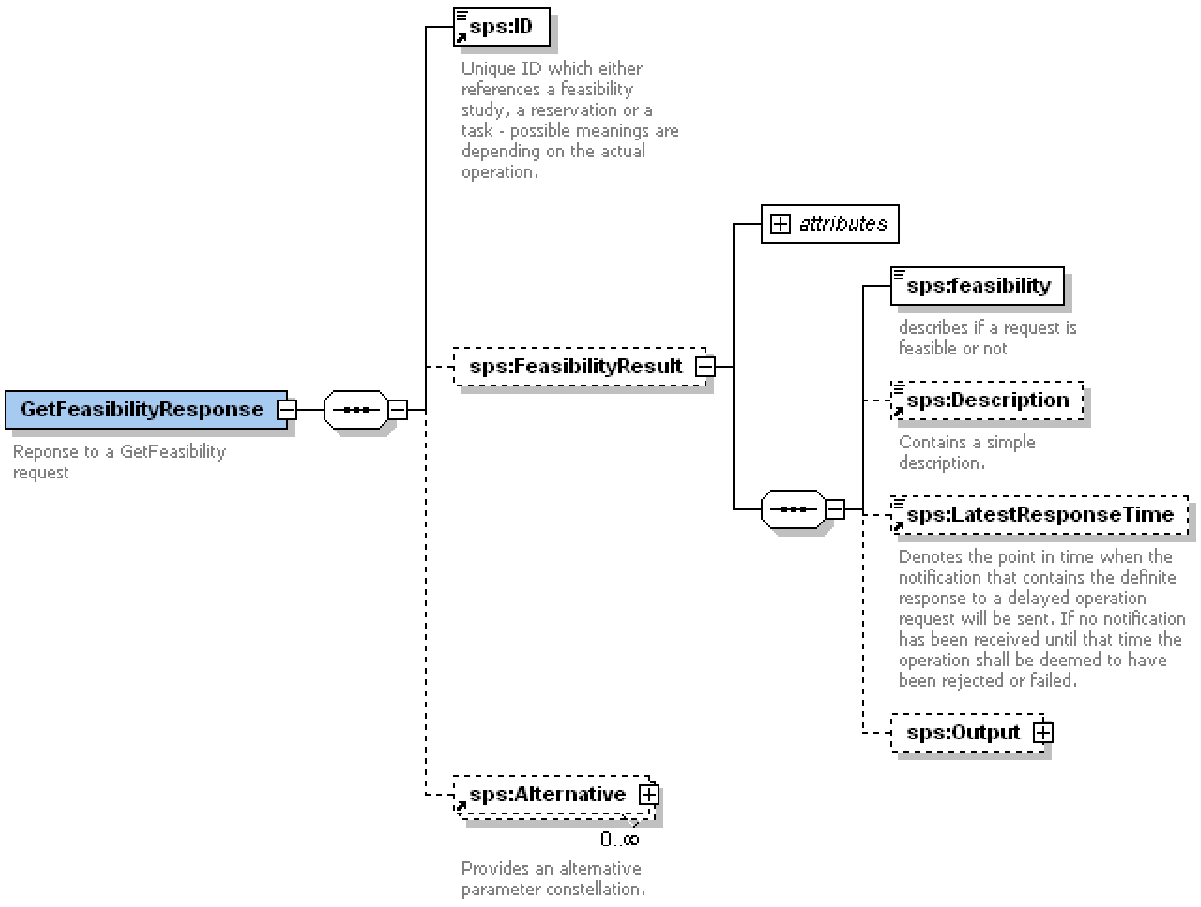

sps:ID
Unique ID which either references a feasibility study, a reservation or a task - possible meanings are depending on the actual operation.
attributes
sps:feasibility
describes if a request is feasible or not
sps:FeasibilityResult
GetFeasibilityResponse
Reponse to a GetFeasibility request
sps:Description
Contains a simple description.
sps:LatestResponseTime
Denotes the point in time when the notification that contains the definite response to a delayed operation request will be sent. If no notification has been received until that time the operation shall be deemed to have been rejected or failed.
sps:Output
sps:Alternative
0..∞
Provides an alternative parameter constellation.

Figure 15-9: - GetFeasibility response

The following table describes the elements of the GetFeasibility response schema:

Table 15-2: Parts of GetFeasibility operation response

Name	Definition	Data type and values	Multiplicity and use
ID	Identifier for the feasibility study	Token	one (mandatory)
Feasibility Result			
feasibility	Identifier for the feasibility status	String, enumerates: "feasible" "not feasible" "response delayed, notification will be sent" "request incomplete" "not feasible, alternatives available"	one (mandatory)
Description	Text description of the response	StringOrRefType	one (optional)
Latest Response Time	GetFeasibility response will be sent until LatestResponseTime at latest. In case that no response is received, the operation shall be evaluated as non-feasible.	gml:TimeInstantType	one (mandatory)
Output / parameters	Encoding should match description in **OutputParameters** DescribeGetFeasibility response (or DescribeSubmit response if DescribeGetFeasibility is not implemented).	List of scenes which may be acquired to cover the regionOfInterest specified by the client	0 to many
alternative	Possible alternative to a given set of parameters will lead to status "not feasible, alternative available". Encoding should match description in **InputParameters** DescribeGetFeasibility response.	One or more value(s) that the user may use as input parameters of a GetFeasibility or a Submit operation.	one to many (optional) describe the alternative proposal by the GS

15.2.6 GetFeasibility response acknowledgment

Schema of the GetFeasibility response acknowledgment:

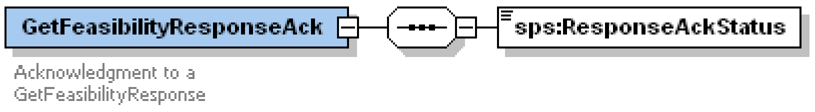

GetFeasibilityResponseAck
Acknowledgment to a GetFeasibilityResponse
sps:ResponseAckStatus

Figure 15-10 - GetFeasibilityResponse acknowledgement

The element ResponseAckStatus is described in § *9.3.3.2 Response acknowledgment status*.

15.2.7 Example

The SPS servers shall implement the GetFeasibility operation request with SOAP and WS-Addressing as in the following example:

```
<!-- Asynchronous -->
<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
xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing">
    <wsa:MessageID>656AH</wsa:MessageID>
    <wsa:ReplyTo>
      <wsa:Address>http://ws.spotimage.com/client_ows/owsclient</wsa:Address>
    </wsa:ReplyTo>
  </soapenv:Header>
  <soapenv:Body>
    <GetFeasibility                service="SPS"                version="1.0.30"
xmlns:gml="http://www.opengis.net/gml"     xmlns="http://www.opengis.net/sps/0"
xmlns:swe="http://www.opengis.net/swe/0"
xmlns:xlink="http://www.w3.org/1999/xlink"
                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/sps/0
file:///D:\users\pmerigot\projets\OGC\dev\doc_conception\NSI\schemas\swe\sps\c
urrent/spsGetFeasibility.xsd">
      <sensorID>urn:ogc:object:feature:Sensor:SpotImage:spot</sensorID>
      <parameters>
        <Parameter parameterID="surveyPeriod">
          <value>
            <swe:TimeRange>2006-12-09T08:00+01:00                    2006-12-
09T08:00+01:00</swe:TimeRange>
          </value>
        </Parameter>
        <Parameter parameterID="regionOfInterest">
          <value>
            <GeometryDefinition>
              <gml:Polygon>
                <gml:exterior>
                  <gml:LinearRing>
                    <gml:posList>35.0 35.0 36.0 35.0 36.0 33.5 35.0 33.5
                                 35.0 35.0</gml:posList>
                  </gml:LinearRing>
                </gml:exterior>
              </gml:Polygon>
            </GeometryDefinition>
          </value>
        </Parameter>
        <Parameter parameterID="imageMode">
          <value>
            <swe:Category>COLOR</swe:Category>
          </value>
        </Parameter>
        <Parameter parameterID="maxCloudCoverage">
          <value>
            <swe:Count>11</swe:Count>
          </value>
        </Parameter>
        <Parameter parameterID="maxSnowCoverage">
          <value>
            <swe:Count>5</swe:Count>
          </value>
        </Parameter>
        <Parameter parameterID="resolution">
          <value>
            <swe:Quantity>2.5</swe:Quantity>
          </value>
        </Parameter>
        <Parameter parameterID="incidenceAngle">
          <value>
            <swe:Category>Unspecified</swe:Category>
          </value>
        </Parameter>
      </parameters>
       <feasibilityLevel>full</feasibilityLevel>
    </GetFeasibility>
  </soapenv:Body>
</soapenv:Envelope>
```

A GetFeasibility response for SPS can look like this encoded in XML:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header>
    <addr:RelatesTo
xmlns:addr="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">656AH</addr:RelatesTo>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <GetFeasibilityResponse                    xmlns="http://www.opengis.net/sps"
xmlns:gml="http://www.opengis.net/gml">
      <ID>fID_215394486</ID>
      <FeasibilityResult>
        <feasibility>feasible</feasibility>
        <Output>
          <Parameter parameterID="scene">
            <swe:component name="id">
              <value>
                <swe:Category>5 066-265 02/07/21 10:05:02 2 J</swe:Category>
              </value>
            </swe:component>
            <swe:component name="satellite">
              <value>
                <swe:Category>SPOT5</swe:Category>
              </value>
            </swe:component>
            <swe:component name="resolution">
              <value>
                <swe:Quantity>2.5</swe:Quantity>
              </value>
            </swe:component>
            <swe:component name="geoLocation">
              <value>
                <gml:Polygon>
                  <gml:exterior>
                    <gml:LinearRing>
                      <gml:posList>-75.0 41.0 -73.0 41.0 -73.0 40.0 -75.0
                                    40.0 -75.0 41.0</gml:posList>
                    </gml:LinearRing>
                  </gml:exterior>
                </gml:Polygon>
              </value>
            </swe:component>
            <swe:component name="successRate">
              <value>
                <swe:Count>70</swe:Count>
              </value>
            </swe:component>
          </Parameter>
        </Output>
      </FeasibilityResult>
    </GetFeasibilityResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

15.2.8 GetFeasibility exceptions

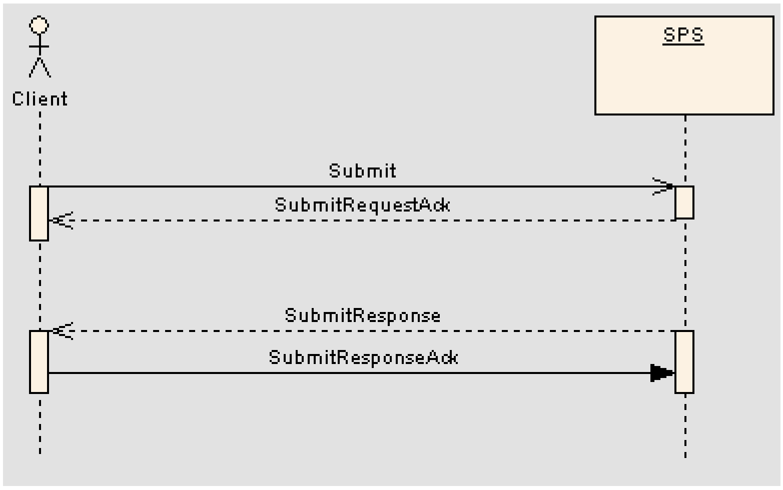
When a SPS server encounters an error while performing a GetFeasibility operation, it shall return an exception report message as specified in Subclause 7.4 of [OGC 05-008].

15.3 Submit operation (mandatory, asynchronous)

15.3.1 Introduction

The Submit operation allows SPS clients to submit a programming request.

Submit uses an asynchronous communication model, as illustrated in the figure below. When the SPS server receives the request, an acknowledgment is sent back by the SPS and the connection is closed. When the SPS has finished the task later on, it will open a new connection (i.e. asynchronously), to send the response back to the client, which itself acknowledges the receive of the response message.



The entire process consists out of 4 messages (described further in the document):

5. the request: *Submit* (§ 15.3.3)
6. the acknowledgment of the request: *SubmitRequestAck* (§ 15.3.4)
7. the response (callback): *SubmitResponse* (§ 15.3.5)
8. the acknowledgment of the response: *SubmitResponseAck* (§ 15.3.6)

15.3.2 EO profile specific

- Acknowledgments.
- New element *DeliveryInformation* as part of the request
- The Submit request contains a new element named *statusNotification***.** Possible values are *once*, *final*, *all*, defined as follow:
    - *once* (or if not specified), the client will receive an immediate and unique SubmitResponse.

  - *Final*, the client will receive the SubmitResponse message when the operation is finished (data is acquired).

  - *All*, the client will receive a new response each time some progress has been done on the submitted task.

- Content of the Submit response of the SPS specification has been dispatched in the Submit acknowledgment and Submit response.

- The response is encapsulated into a new element *ProgressReport* shared with the GetStatus response (described in paragraph 9.3.4).

- The response contains an *Output* element allowing the service to tell the client which scene has been acquired.

15.3.3 Submit request

Schema of the Submit request:

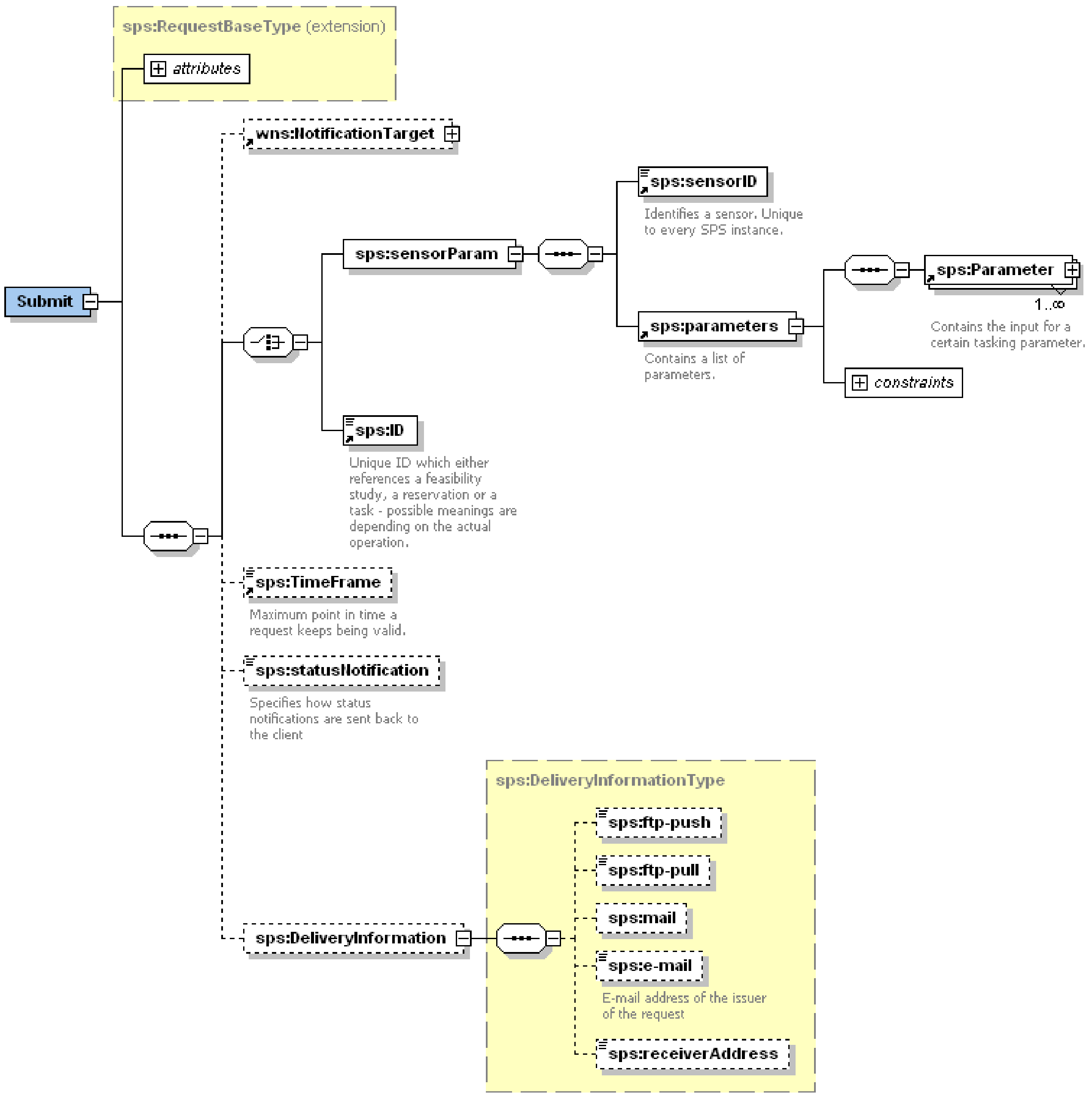

sps:RequestBaseType (extension)
attributes
wns:NotificationTarget
sps:sensorID
Identifies a sensor. Unique to every SPS instance.
sps:sensorParam
sps:Parameter
1..∞
Contains the input for a certain tasking parameter.
Submit
sps:parameters
Contains a list of parameters.
constraints
sps:ID
Unique ID which either references a feasibility study, a reservation or a task - possible meanings are depending on the actual operation.
sps:TimeFrame
Maximum point in time a request keeps being valid.
sps:statusNotification
Specifies how status notifications are sent back to the client
sps:DeliveryInformationType
sps:ftp-push
sps:ftp-pull
sps:mail
sps:DeliveryInformation
sps:e-mail
E-mail address of the issuer of the request
sps:receiverAddress

Figure 15-11: - Submit request schema

The following table describes the elements of the Submit schema:

Tag Name	Tag Description	Multiplicity and use
notificationTarget	From ([NR13]): Defines the WNS that has to be used to notify the client about the request results	One (alternative to notification)
sensorParam	container element contains sensorID and parameters elements Type: complex	Alternative to feasibilityID
sensorParam / parameters	Input parameter values. Encoding should match description in **InputParameters** DescribeSubmit response Complex See § 8	One, mandatory only instead of ID element
ID	Identifier of the feasibility study ID that was provided by SPS on behalf of a GetFeasibility request String feasibilityID	one (mandatory) only instead of parameters element
timeFrame	From ([NR13]): Maximum point in time the request keeps valid.	One, optional
statusNotification	Specifies how many notifications are sent back to the client (see§ EO profile specific). Type: enumerated string Possible values are: once (default), final, all	One, optional
DeliveryInformation	Information about how the products of the programming request have to be delivered. See § 8.5	One, optional

Table 15-3: Submit elements descriptions

15.3.4 Submit request acknowledgement

Schema of the acknowledgement of the Submit request:

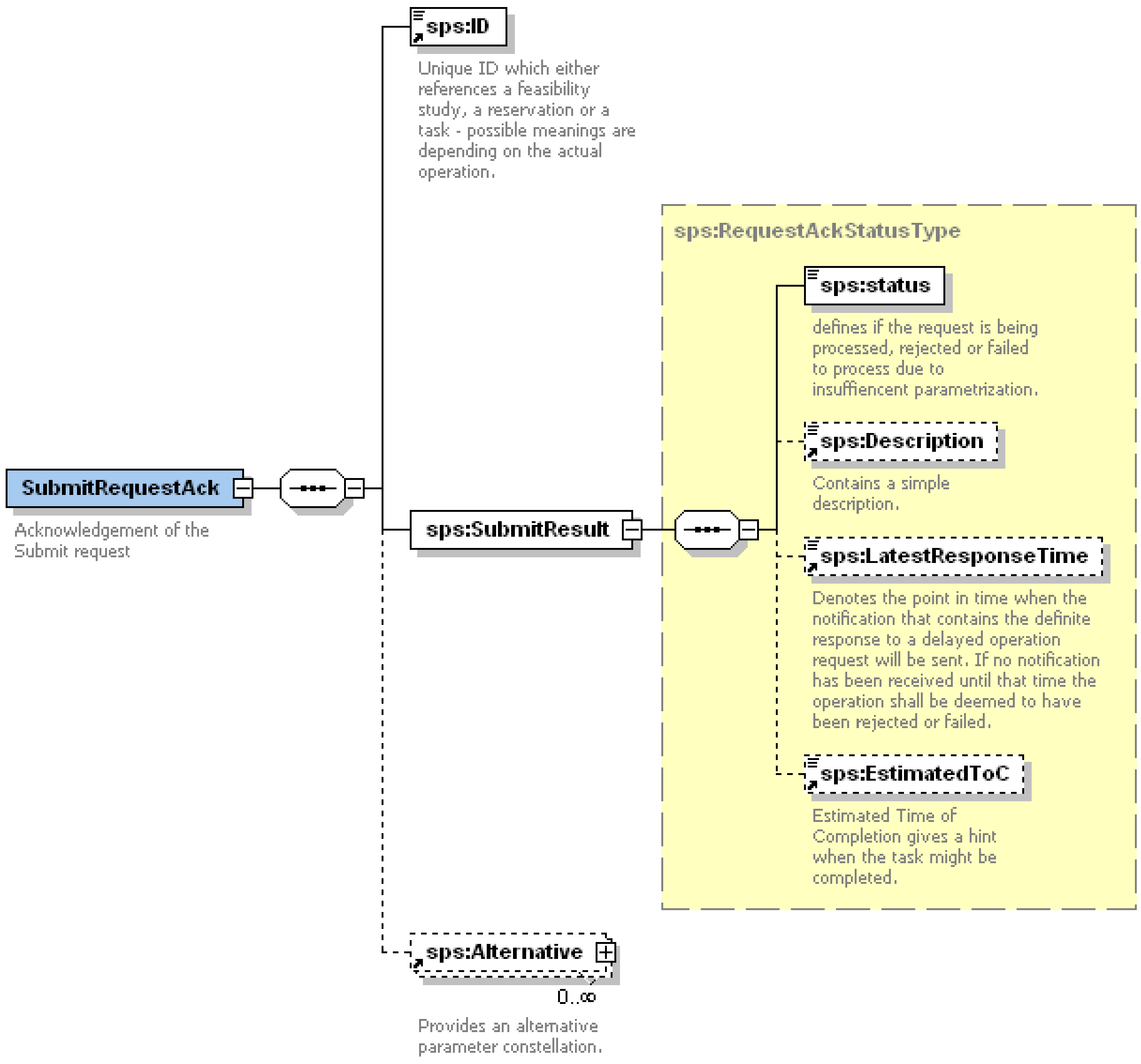

sps:ID
Unique ID which either references a feasibility study, a reservation or a task - possible meanings are depending on the actual operation.
SubmitRequestAck
Acknowledgement of the Submit request
sps:SubmitResult
sps:RequestAckStatusType
sps:status
defines if the request is being processed, rejected or failed to process due to insuffiencent parametrization.
sps:Description
Contains a simple description.
sps:LatestResponseTime
Denotes the point in time when the notification that contains the definite response to a delayed operation request will be sent. If no notification has been received until that time the operation shall be deemed to have been rejected or failed.
sps:EstimatedToC
Estimated Time of Completion gives a hint when the task might be completed.
sps:Alternative
0..∞
Provides an alternative parameter constellation.

Figure 15-12 - SubmitRequestAck schema

The following table describes the elements of the SubmitRequestAck schema:

Table 15-4: Parts of Submit operation response

Name	Definition	Data type and values	Multiplicity and use
ID	Identifier for this task, needed for subsequent update requests (taskID).	token	one (mandatory)
SubmitResult	See § 9.3.3.1 Request acknowledgment status	RequestAckStatus	
Alternative	Provides alternatives if the sensors are not taskable as requested	complex: InputParameter	one to many (optional)

15.3.5 Submit response

Schema of the Submit response:

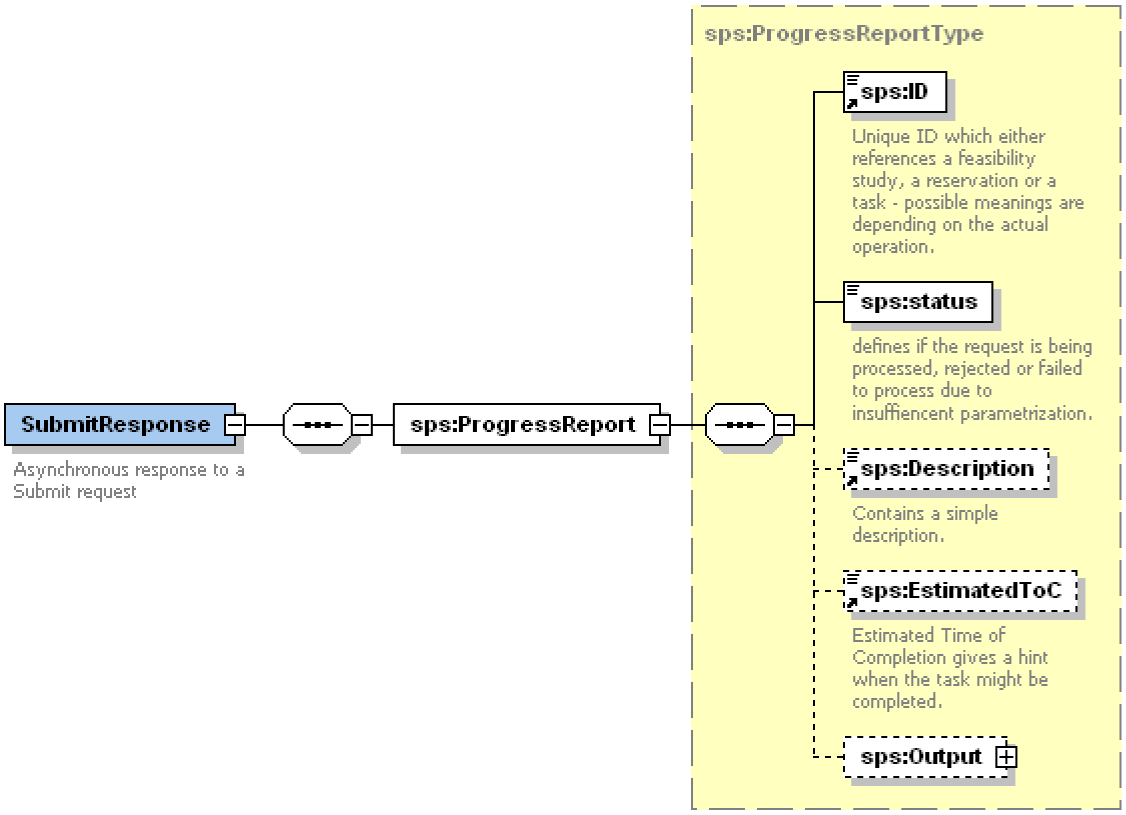

sps:ProgressReportType
sps:ID
Unique ID which either references a feasibility study, a reservation or a task - possible meanings are depending on the actual operation.
sps:status
defines if the request is being processed, rejected or failed to process due to insuffiencent parametrization.
sps:Description
Contains a simple description.
sps:EstimatedToC
Estimated Time of Completion gives a hint when the task might be completed.
sps:Output
SubmitResponse
Asynchronous response to a Submit request
sps:ProgressReport

Figure 15-13 - SubmitResponse schema

The element ProgressReport is described in § *9.3.4*.

Note when sending a Submit request: if the user sets the value of StatusNotification to *All*, a submit response is sent by the SPS each time a scene is acquired. This is particularly useful in case of an acquisition per week during one year for example. In this case the *Output* element of *ProgressReport* contains the description of the new acquired scene only.

15.3.6 Submit response acknowledgment

Schema of the acknowledgment of the Submit response:

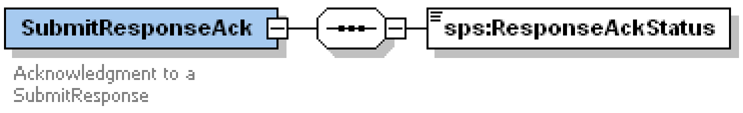


Figure 15-14 - SubmitResponseAck schema

The element ResponseAckStatus is described in § *9.3.3.2 Response acknowledgment status*.

15.3.7 Example

Example of Submit request using of SOAP and WS-Addressing:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Asynchronous -->
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header
xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing">
    <wsa:MessageID>661JH</wsa:MessageID>
    <wsa:ReplyTo>
<wsa:Address>http://ws.spotimage.com/client_ows/owsclient</wsa:Address>
    </wsa:ReplyTo>
  </soapenv:Header>
  <soapenv:Body>
    <Submit  service="SPS"  version="1.0.30"  xmlns="http://www.opengis.net/sps"
xmlns:swe="http://www.opengis.net/swe" xmlns:gml="http://www.opengis.net/gml">
      <sensorParam>
<sensorID>urn:ogc:object:feature:Sensor:SpotImage:spot_ows4</sensorID>
        <parameters>
          <Parameter parameterID="surveyPeriod">
            <swe:component name="start-of-period">
              <value>
                <swe:Time>2006-11-20T08:00+01:00</swe:Time>
              </value>
            </swe:component>
            <swe:component name="end-of-period">
              <value>
                <swe:Time>2006-12-30T08:00+01:00</swe:Time>
              </value>
            </swe:component>
          </Parameter>
          <Parameter parameterID="regionOfInterest">
            <value>
              <gml:Polygon>
                <gml:exterior>
                  <gml:LinearRing>
                    <gml:posList>-75.0 41.0 -73.0 41.0 -73.0 40.0 -75.0 40.0
                                 -75.0 41.0</gml:posList>
                  </gml:LinearRing>
                </gml:exterior>
              </gml:Polygon>
            </value>
          </Parameter>
          <Parameter parameterID="imageMode">
            <value>
              <swe:Category>COLOR</swe:Category>
            </value>
          </Parameter>
          <Parameter parameterID="maxCloudCoverage">
            <value>
              <swe:Count>10</swe:Count>
            </value>
          </Parameter>
          <Parameter parameterID="maxSnowCoverage">
            <value>
              <swe:Count>10</swe:Count>
            </value>
          </Parameter>
          <Parameter parameterID="resolution">
            <value>
              <swe:Quantity>2.5</swe:Quantity>
            </value>
          </Parameter>
          <Parameter parameterID="incidenceAngle">
            <value>
              <swe:Category>Unspecified</swe:Category>
            </value>
          </Parameter>
          <Parameter parameterID="country">
            <swe:component name="countryCode">
              <value>
                <swe:Category>UM</swe:Category>
              </value>
            </swe:component>
            <swe:component name="countryName">
              <value>
                <swe:Category>UNITED        STATES        MINOR        OUTLYING
ISLANDS</swe:Category>
              </value>
            </swe:component>
          </Parameter>
          <Parameter parameterID="placeName">
            <value>
              <swe:Category>New-York</swe:Category>
            </value>
          </Parameter>
          <Parameter parameterID="comment">
            <value>
              <swe:Category>New-York City</swe:Category>
            </value>
          </Parameter>
          <Parameter parameterID="context">
            <value>
              <swe:Category>DEMO</swe:Category>
            </value>
          </Parameter>
        </parameters>
      </sensorParam>
    </Submit>
  </soapenv:Body>
</soapenv:Envelope>
```

Example of Submit request acknowledgment:

```
<SubmitRequestAck xmlns="http://www.opengis.net/sps"
xmlns:gml="http://www.opengis.net/gml">
    <ID>433</ID>
    <SubmitResult>
        <status>confirmed</status>
        <LatestResponseTime>
            <gml:TimeInstant>
                <gml:timePosition>2005-10-05T16:15:00</gml:timePosition>
            </gml:TimeInstant>
        </LatestResponseTime>
        <estimatedToC>
            <gml:TimeInstant>
                <gml:timePosition>2005-10-05T16:55:00+02:00</gml:timePosition>
            </gml:TimeInstant>
        </estimatedToC>
    </SubmitResult>
</ SubmitRequestAck >
```

16 GetStatus operation (optional, synchronous)

16.1 Introduction

The GetStatus operation allows a client to get information about the current status of a specific task.

16.2 EO profile specific

- The GetStatus request contains a new element named **DateFrom**. This optional element may be useful in case a repetitive acquisition (example: once a week) is made over a long period. By providing the current date as payload of the **DateFrom** element, the response will contain the status of the last acquired image only. If no DateFrom would be defined, the whole list of images acquired since the beginning of the acquisition period would be returned.

- The response is encapsulated into a new element *ProgressReport* shared with the Submit response (describe in paragraph *9.3.4*).

- The GetStatusResponse contains an Output element allowing the service to tell the client which scenes have been acquired (cf Figure 16-2: - GetStatus response schema).

16.3 GetStatus operation schemas

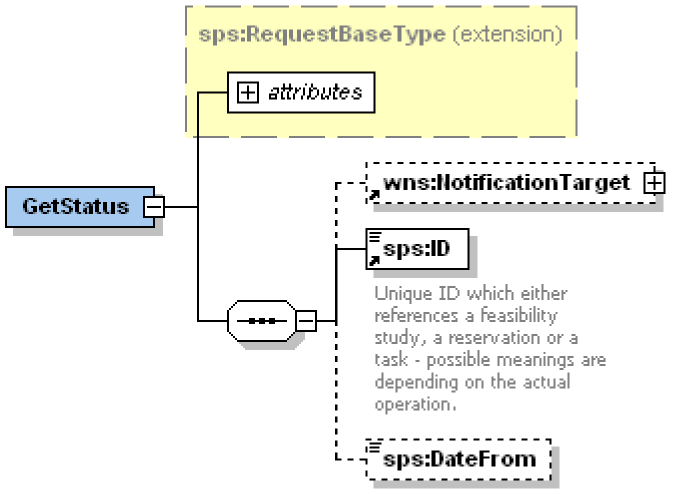

sps:RequestBaseType (extension)
attributes
GetStatus
wns:NotificationTarget
sps:ID
Unique ID which either references a feasibility study, a reservation or a task - possible meanings are depending on the actual operation.
sps:DateFrom

Figure 16-1: - GetStatus request schema

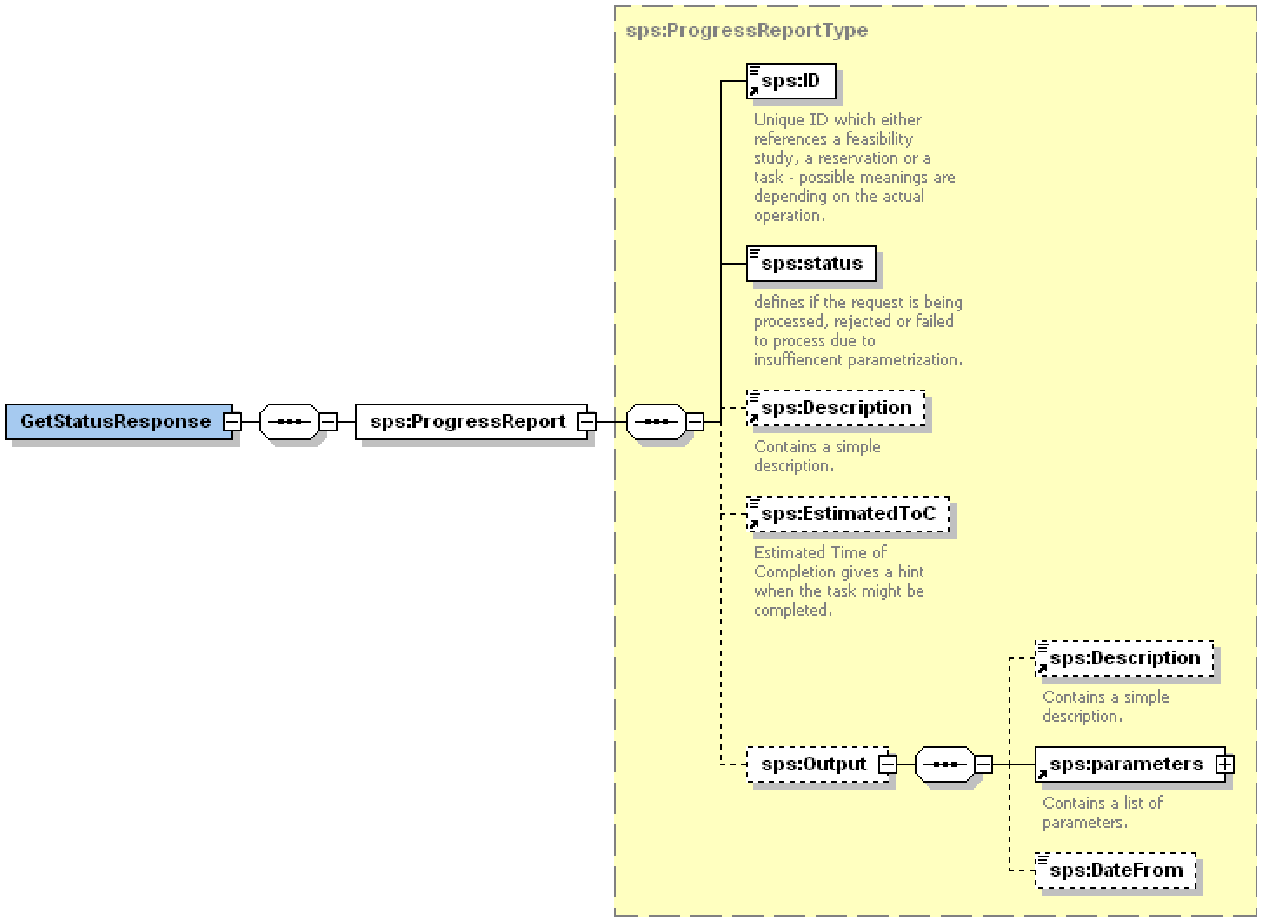

sps:ProgressReportType
sps:ID
Unique ID which either references a feasibility study, a reservation or a task - possible meanings are depending on the actual operation.
sps:status
defines if the request is being processed, rejected or failed to process due to insuffiencent parametrization.
sps:Description
Contains a simple description.
sps:EstimatedToC
Estimated Time of Completion gives a hint when the task might be completed.
sps:Description
Contains a simple description.
sps:Output
sps:parameters
Contains a list of parameters.
sps:DateFrom
GetStatusResponse
sps:ProgressReport

Figure 16-2: - GetStatus response schema

16.4 GetStatus operation request

16.4.1 Parameters

Table 16-1: Parameters in GetStatus operation

Name [a]	Definition	Data type and values	Multiplicity and use
service	Service type identifier	Character String type, not empty Value is OWS type abbreviation "SPS"	One (mandatory)
version	Specification version for operation	Character String type, not empty Equals the number of this document.	One (mandatory)
notification Target	WNS data that should be used by the SPS server	HMA context: not used (see § 9.3.2)	one (optional)
ID	Identifier of the task (taskID)	Token	one (mandatory)
DateFrom	see above § 16.2	Date	one (optional)
a The name capitalization rules being used here are specified in Subclause 11.6.2 of [OGC 05-008].			

16.4.2 GetStatus request XML encoding

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<GetStatus xmlns="http://www.opengis.net/sps" service="SPS" version="0.0.30">
  <ID>433</ID>
</GetStatus>
```

16.5 GetStatus operation response

The element ProgressReport is described in § 9.3.4.

16.6 GetStatus exceptions

When a SPS server encounters an error while performing a GetStatus operation, it shall return an exception report message as specified in Subclause 7.4 of [OGC 05-008].

17 Update operation (optional, synchronous)

17.1 Introduction

The Update operation allows a client to update a previously submitted task.

17.2 Update operation schemas

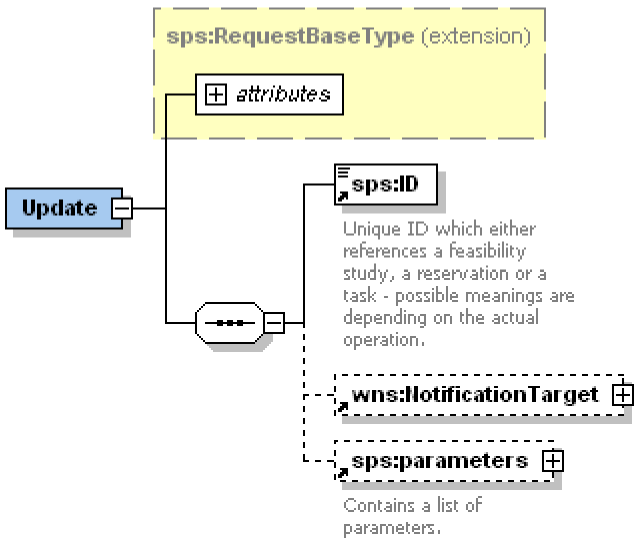

sps:RequestBaseType (extension)
attributes
Update
sps:ID
Unique ID which either references a feasibility study, a reservation or a task - possible meanings are depending on the actual operation.
wns:NotificationTarget
sps:parameters
Contains a list of parameters.

Figure 17-1 - Update request schema

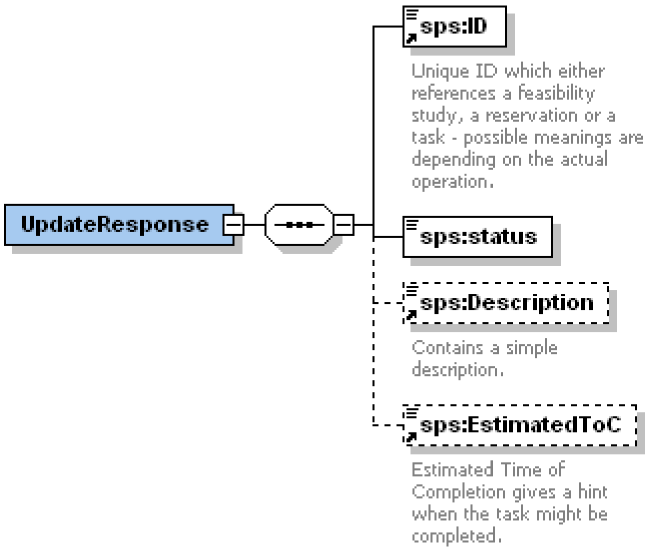

sps:ID
Unique ID which either references a feasibility study, a reservation or a task - possible meanings are depending on the actual operation.
UpdateResponse
sps:status
sps:Description
Contains a simple description.
sps:EstimatedToC
Estimated Time of Completion gives a hint when the task might be completed.

Figure 17-2 - Update response schema

17.3 Update operation request

17.3.1 Parameters

Table 17-1: Parameters in UpdateRequest operation

Name [a]	Definition	Data type and values	Multiplicity and use
service	Service type identifier	Character String type, not empty Value is OWS type abbreviation "SPS"	One (mandatory)
version	Specification version for operation	Character String type, not empty Equals the number of this document	One (mandatory)
ID	Identifier for the task that shall be updated (taskID)	token	one (mandatory)
parameters	update parameterization for the task	complex	one (optional)
a The name capitalization rules being used here are specified in Subclause 11.6.2 of [OGC 05-008].			

17.3.2 Update request XML encoding

All SPS servers shall implement HTTP POST transfer of the UpdateRequest operation, using XML encoding only. The following schema fragment specifies the contents and structure of a UpdateRequest operation encoded in XML.

EXAMPLE

```
<Update                                      xmlns="http://www.opengis.net/sps"
xmlns:swe="http://www.opengis.net/swe" service="SPS" version="0.0.30">
   <ID>433</ID>
   <parameters>
      <Parameter parameterID="pan">
         <value>
            <swe:Category>30</swe:Category>
         </value>
      </Parameter>
      <Parameter parameterID="tilt">
         <value>
            <swe:Category>5</swe:Category>
         </value>
      </Parameter>
   </parameters>
</Update>
```

17.4 Update operation response

17.4.1 Parameters

Table 17-2: Parts of Update operation response

Name	Definition	Data type and values	Multiplicity and use
ID	Identifies the updated task, provided by SPS server (taskID)	token	one (mandatory)
status	Status of the Update request.	Enumerated String. Possible values are: • Confirmed • Rejected • incomplete	one (mandatory)
Description	additional continuous text	String	one (optional)
estimatedToC	Estimated Time of Completion for this task	DateTime	one (optional)

17.4.2 Update response example

A Update operation response for SPS can look like this encoded in XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<UpdateResponse xmlns="http://www.opengis.net/sps">
  <ID>433</ID>
  <status>confirmed</status>
</UpdateResponse>
```

17.5 Update exceptions

When a SPS server encounters an error while performing an Update operation, it shall return an exception report message as specified in Subclause 7.4 of [OGC 05-008]. The allowed standard exception codes shall include those listed in the following table. For each listed exceptionCode, the contents of the “locator” parameter value shall be as specified in the right column.

Table 17-3: Exception codes for UpdateRequest operation

exceptionCode value	Meaning of code	“locator” value
OperationNotSupported	Request is for an operation that is not supported by this server	Name of operation not supported
MissingParameterValue	Operation request does not include a parameter value, and this server did not declare a default value for that parameter	Name of missing parameter
InvalidParameterValue	Operation request contains an invalid parameter value	Name of parameter with invalid value
NoApplicableCode	No other exceptionCode specified by this service and server applies to this exception	None, omit “locator” parameter
TaskIDExpired	ID that has been issued by the client is no longer supported by the service	None, omit 'locator' parameter
InvalidRequest	Request is not conform to the schema for this operation	Exception message generated by validator

18 Cancel operation (optional, synchronous)

18.1 Introduction

The Cancel operation cancels a previously requested task.

18.2 Cancel operation schemas

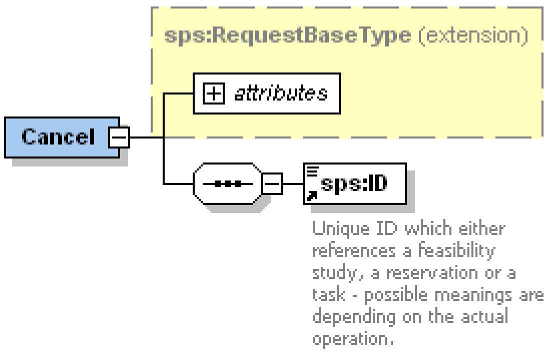

sps:RequestBaseType (extension)
attributes
Cancel
sps:ID
Unique ID which either references a feasibility study, a reservation or a task - possible meanings are depending on the actual operation.

Figure 18-1 - Cancel request schema

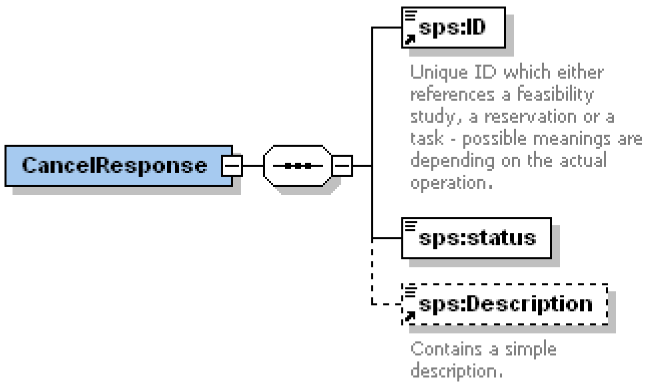

sps:ID
Unique ID which either references a feasibility study, a reservation or a task - possible meanings are depending on the actual operation.
CancelResponse
sps:status
sps:Description
Contains a simple description.

Figure 18-2 - Cancel response schema

18.3 Cancel operation request

Table 18-1: Parameters in Cancel operation request

Name [a]	Definition	Data type and values	Multiplicity and use
service	Service type identifier	Character String type, not empty Value is OWS type abbreviation "SPS"	One (mandatory)
request	Operation name	Character String type, not empty Value is operation name "Cancel"	One (mandatory)
version	Specification version for operation	Character String type, not empty Equals the number of this document	One (mandatory)
ID	Identifies the task to be cancelled	token	one (mandatory)
a The name capitalization rules being used here are specified in Subclause 11.6.2 of [OGC 05-008].			

An example Cancel operation request XML encoded for HTTP POST is:

```
<Cancel xmlns="http://www.opengis.net/sps" service="SPS" version="0.0.30">
   <ID>433</ID>
</Cancel>
```

18.4 Cancel operation response

Table 18-2: Parts of Cancel operation response

Name	Definition	Data type and values	Multiplicity and use
ID	Identifies the task	token	one (mandatory)
status		String Possible values are: - confirmed - rejected	one (mandatory)
description	further information in continuous text	String	one (optional)

A Cancel operation response for SPS can look like this encoded in XML:

```
<CancelResponse xmlns="http://www.opengis.net/sps">
   <ID>433</ID>
   <status>confirmed</status>
</CancelResponse>
```

18.5 Cancel exceptions

When a SPS server encounters an error while performing a Cancel operation, it shall return an exception report message as specified in Subclause 7.4 of [OGC 05-008].

19 DescribeResultAccess operation (mandatory, synchronous)

19.1 Introduction

The DescribeResultAccess operation allows SPS clients to retrieve information where the observed data can be accessed from. For the EO Application profile, the response should either point to an OGC Web Service (desirable option) or to a datastore accessible using FTP. The link to the acquired data is needed to close the acquisition loop and provide an access to the data or a preview of this data to validate the image acquired.

19.2 EO profile specific

- The DescribeResultAccess request contains a new element named **DateFrom** (see schema below). This optional element may be useful in case a repetitive acquisition (example: once a week) is made over a long period. By providing the current date as payload of the **DateFrom** element, the response will contain the status of the last acquired image only. If no DateFrom would be defined, the whole list of images acquired since the beginning of the acquisition period would be returned.

- The DescribeResultAccess response contains a new element productId in order to get the products resulting from the programming request from a separated service like the Ordering service ([NR12]).

19.3 DescribeResultAccess operation schemas

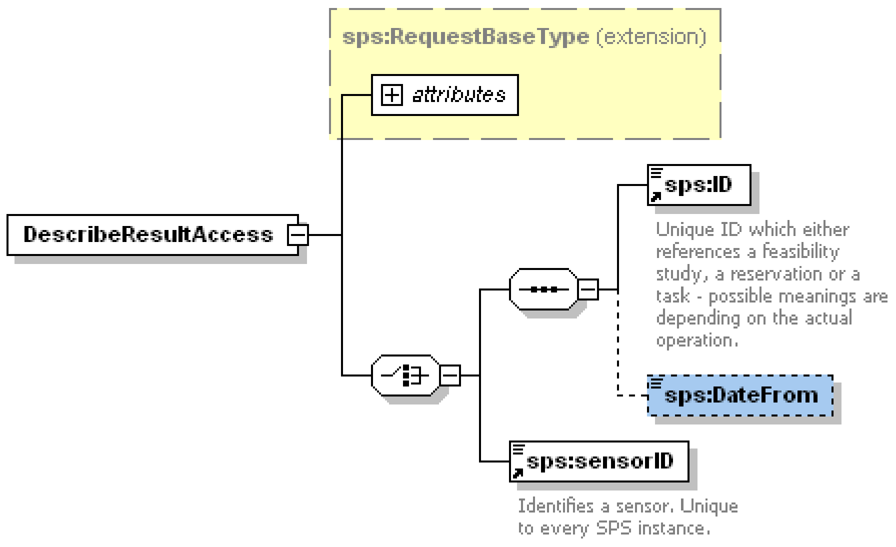

sps:RequestBaseType (extension)
attributes
DescribeResultAccess
sps:ID
Unique ID which either references a feasibility study, a reservation or a task - possible meanings are depending on the actual operation.
sps:DateFrom
sps:sensorID
Identifies a sensor. Unique to every SPS instance.

Figure 19-1: - DescribeResultAccess request schema

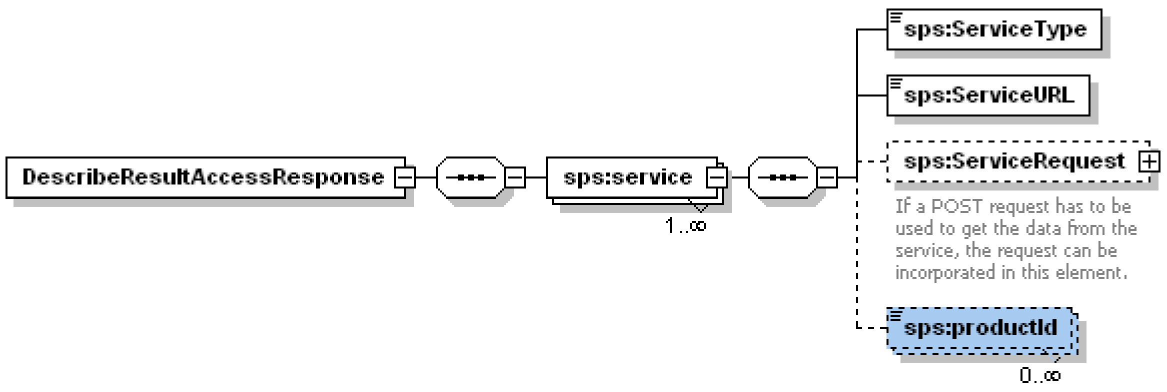

DescribeResultAccessResponse
sps:service
1..∞
sps:ServiceType
sps:ServiceURL
sps:ServiceRequest
If a POST request has to be used to get the data from the service, the request can be incorporated in this element.
sps:productId
0..∞

Figure 19-2: - DescribeResultAccess response schema

19.4 DescribeResultAccess operation request

Table 19-1: Parameters in DescribeResultAccess operation request

Name [a]	**Definition**	**Data type and values**	**Multiplicity and use**
service	Service type identifier	Character String type, not empty Value is OWS type abbreviation "SPS"	One (mandatory)
request	Operation name	Character String type, not empty Value is operation name "DescribeResultAccess"	One (mandatory)
version	Specification version for operation	Character String type, not empty Equals the number of this document	One (mandatory)
ID	Identifies task (taskID)	Token	one (mandatory; if sensorID is not used)
sensorID	If specified the service will return the service(s) which will serve the data when acquired[b].	Token	one (mandatory; if ID is not used)
DateFrom	see 19.2	Date	optional
a The name capitalization rules being used here are specified in Subclause 11.6.2 of [OGC 05-008]. b Allows the client to know if it can deal with the service serving the data before sending a submit request.			

19.5 DescribeResultAccess operation response

Table 19-2: Parts of DescribeResultAccess operation response

Name	Definition	Data type and values	Multiplicity and use
service	Element containing the type and URL of the OGC Web service that provides access to the observed data	complex,.	one to many (mandatory)
ServiceType	Defines the type of the OGC Web Service	String example: WCS HMA context: CSW	one (mandatory)
ServiceURL	Defines the URL of the OGC Web service that provides access to the observed data	String HMA context: getRecordById	one (mandatory)
Service Request	In case of data is available through the service via a POST request, this field contains the POST request.		
productId	Identifier of the product.		
a			

A DescribeResultAccess operation response for SPS can look like this encoded in XML:

```
<?xml version="1.0"?>
<DescribeResultAccessResponse xmlns="http://www.opengis.net/sps">
  <service>
    <ServiceType>WCS</ServiceType>
    <ServiceURL>http://ws.spotimage.fr/wcs/coverage/SCENE_5_260-
334_0_04-12-30_03-56-15_2_J</ServiceURL>
  </service>
</DescribeResultAccessResponse>
```

19.6 DescribeResultAccess exceptions

When a SPS server encounters an error while performing a DescribeResultAccess operation, it shall return an exception report message as specified in Subclause 7.4 of [OGC 05-008].

20 Implementation guidance

The following section provides some help to developers when setting up a programming service instance that complies with this interface specification. Any information provided here is non-normative. It will be extended and described in more detail in future editions.

20.1.1 Distributed Programming Requests implementation

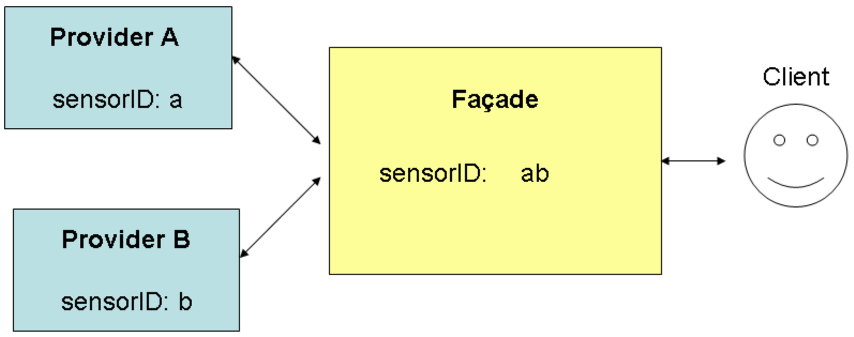
The Programming Service operations have been defined to support also a multi-provider scenario where the client is allowed to build and submit programming requests involving Earth Observation products managed by different providers (e.g. ESA Multi-Mission Ground Segment, SPOT, Radarsat-2 CSA Ground Segment). In this scenario, we consider different Programming Service instances with different roles:

- **Façade Programming Service Instance**, which is the intermediary element in charge of providing the clients a transparent access to the different providers and to orchestrate the access to them.
- **Delegated Programming Service Instances**, which are the services instances running in the providers' environment that are in charge of effectively carrying out the programming requests.

In the following sections the interactions occurring between the different service instances for executing the Programming Service operations are described.

20.1.1.1 sensorID scenario

As describe in the paragraph 8.4, a sensorID may represent a combination of sensors. In the context of multi-provider, the Façade Programming Service may provide new sensorID representing the combination of sensorID provided by different Programming Service Instances. In the example below, the Façade provides a new sensorID (***ab***) representing the combination of the sensorID ***a*** (from Provider A) and the sensorID ***b*** (from Provider B):



This allows the Façade to provide to the Client a unique sensorID for all radar or all optical missions, for example.

20.1.1.2 DescribeGetFeasibility scenario

The following scenario describes the individual steps to provide all necessary programming parameters to a client. The programming parameters are required to task a sensor appropriately.

- The client requests the definition of the programming parameters by giving a sensorID as input parameter. This sensorID provided by the façade may represent a set of sensorID coming from different missions (see § 20.1.1.1).

- The façade Programming Service instance retrieves the information about the programming services able to manage the required sensors. This step becomes more complex in case of loosely defined sensors (not all attributes of the sensorIdentifier element are specified but e.g. only the satellite or only the instrument is specified). In this case the Façade Programming Service has to choose one mission / sensor between the ones available that match the request.

- The programming parameters are requested from the specific providers and then sent back to the client.

20.1.1.3 Get Feasibility scenario

This scenario explains the steps performed by the different service instances to verify the feasibility of a programming request.

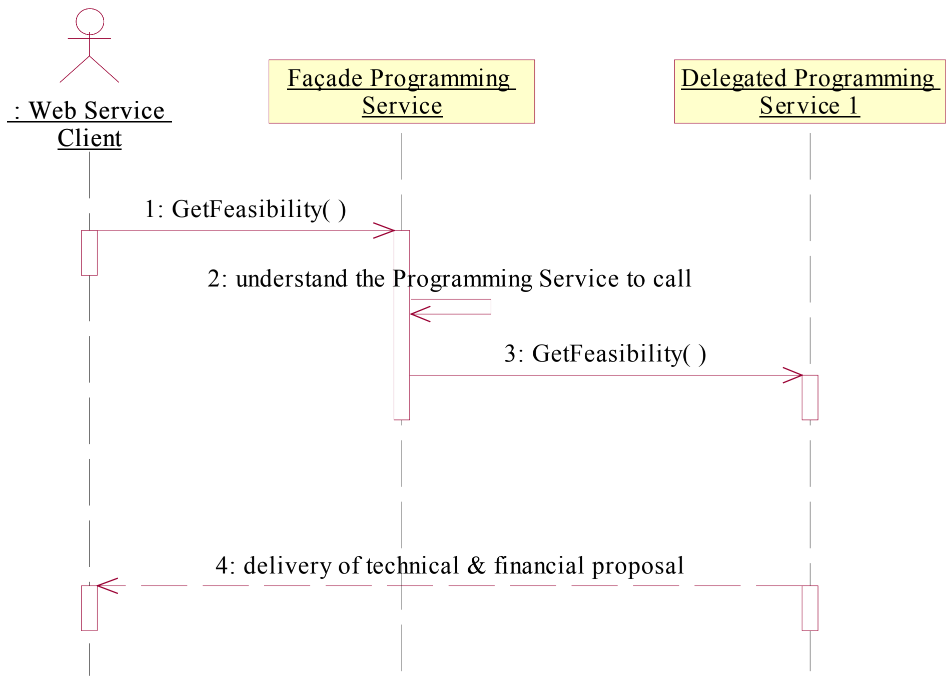

: Web Service Client
Façade Programming Service
Delegated Programming Service 1
1: GetFeasibility( )
2: understand the Programming Service to call
3: GetFeasibility( )
4: delivery of technical & financial proposal

Figure 20-1: Get Feasibility Scenario

- The client prepares a programming request by providing values for those parameters that have been described by the DescribeGetFeasibility operation response, e.g. all those necessary to run a feasibility analysis.
- The façade Programming Service instance retrieves the information about the programming service able to manage the specified sensor (considering also loosely specified sensors);
- The specific parts of the GetFeasibility request is forwarded to delegated Programming Service instances as identified in the previous step.
- The response provided by the delegated programming service is gathered by the Façade Programming Service and sent back to the client. Possibly delayed analysis is sent to the client by the delegated programming service directly. The service uses the notification information specified in the GetFeasibility request
- The feasibilityID returned by the programming service together with the list of related programming parameters is stored locally for being re-used at a later stage (usually to run the Submit operation).
- The technical and financial proposals are delivered later on to the users.

20.1.1.4 Submit scenario

This scenario explains the steps performed by the different service instances when a programming request is submitted by the client.

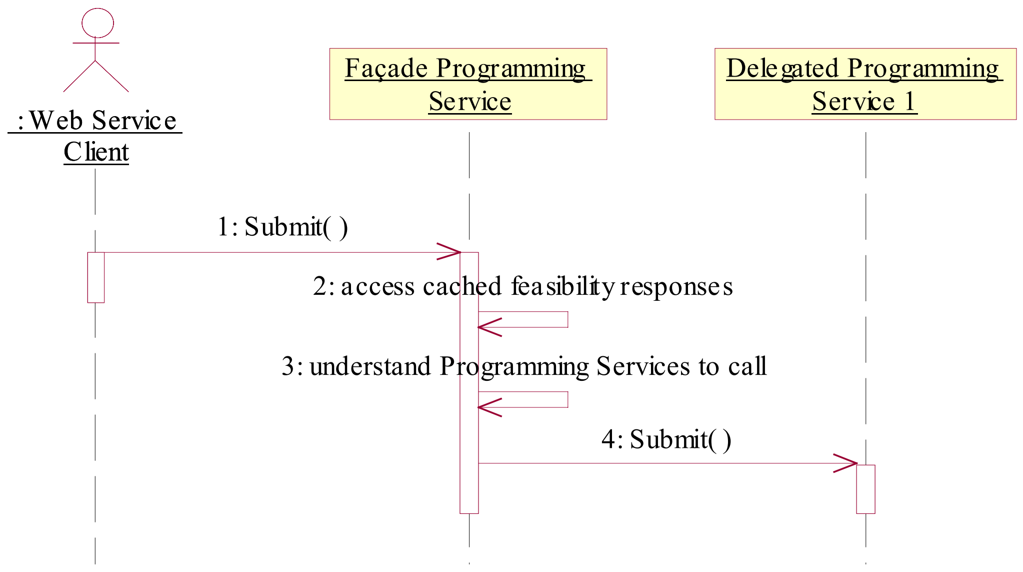

:Web Service Client
Façade Programming Service
Delegated Programming Service 1
1: Submit( )
2: access cached feasibility responses
3: understand Programming Services to call
4: Submit( )

Figure 20-2: Submit Scenario.

- The client has prepared a programming request including either a task that has been already checked (i.e. the feasibility study returned true or valid alternatives) or one that was not preceded by a feasibility study. In the first case, the feasibilityID of that task can be specified, whereas in the second case, all tasking parameters have to be specified.

- In case the task is specified by the feasibilityID, the Façade can either send the request as it is to the delegated service or can retrieve the locally stored parameters and send a Submit request fully specified. The latter can be useful in case of remote services not caching the feasibility results.

- In case the task is specified by parameters for sensor tasking, the Façade Programming Service instance identifies the corresponding Programming Service instance able to manage that sensor and then send to it a Submit request including all the parameters.

- The delegated service returns back the taskID related to the submitted request. The façade instance forwards this information to the client and stores it locally to handle future client requests, e.g. operations like Cancel(TaskID).

20.1.1.5 Get Status scenario

This scenario explains the steps performed by the different service instances when the client asks the status of previously submitted programming requests.

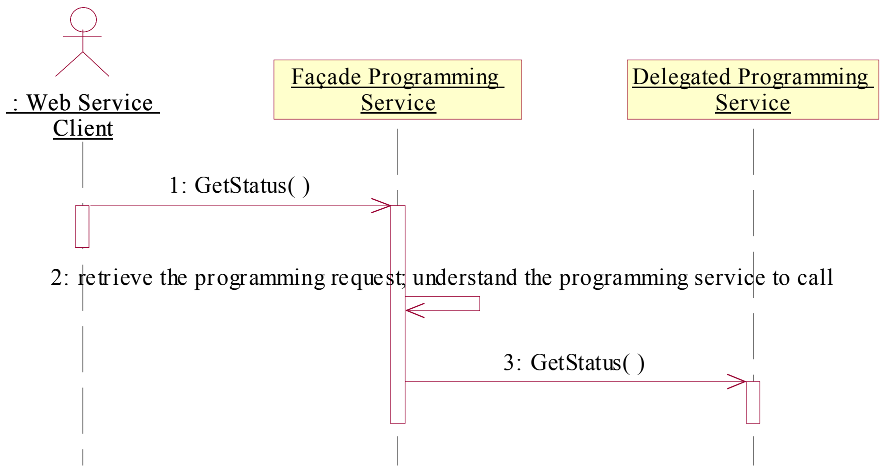

: Web Service Client
Façade Programming Service
Delegated Programming Service
1: GetStatus( )
2: retrieve the programming request; understand the programming service to call
3: GetStatus( )

Figure 20-3: Get Programming Status Scenario.

- The Façade Service instance extracts the address of the Programming service that received the task specified in the input parameter.
- A GetStatus request is sent to the corresponding delegated service. The received status is forwarded to the client and stored locally to handle future client requests.

20.1.1.6 Cancel scenario

This scenario explains the steps performed by the different service instances when the client asks the cancellation of a submitted programming request.

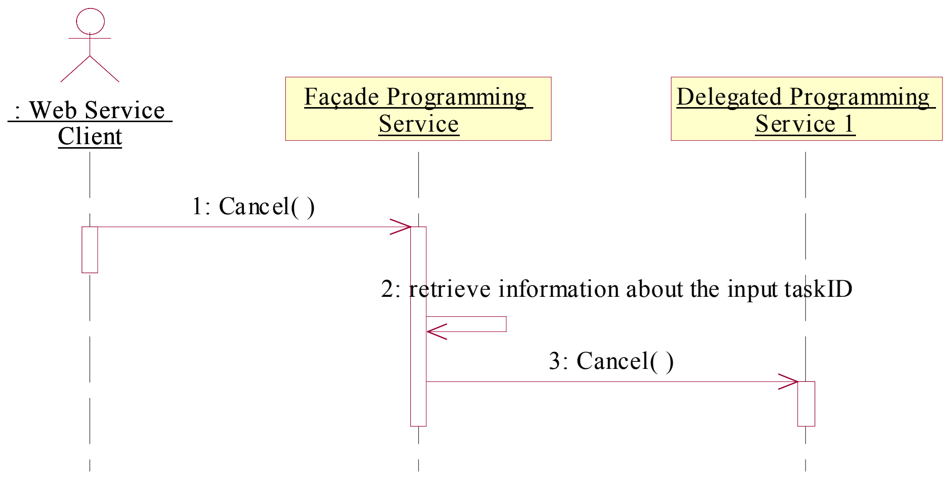

: Web Service Client
Façade Programming Service
Delegated Programming Service 1
1: Cancel( )
2: retrieve information about the input taskID
3: Cancel( )

Figure 20-4: Cancel scenario.

- The client requests the cancellation of a specified task providing the taskID;
- The Façade instance extracts the data related to that task by accessing the local database. In this way it is possible to find the Programming service that is actually managing the task.
- The identified Programming service is requested to cancel the task. Responses will be sent according to the operation specifications described in this specification.

20.1.1.7 DescribeResultAccess scenario

This scenario explains the interactions occurring between the different service instances when the client calls the DescribeResultAccess operation.

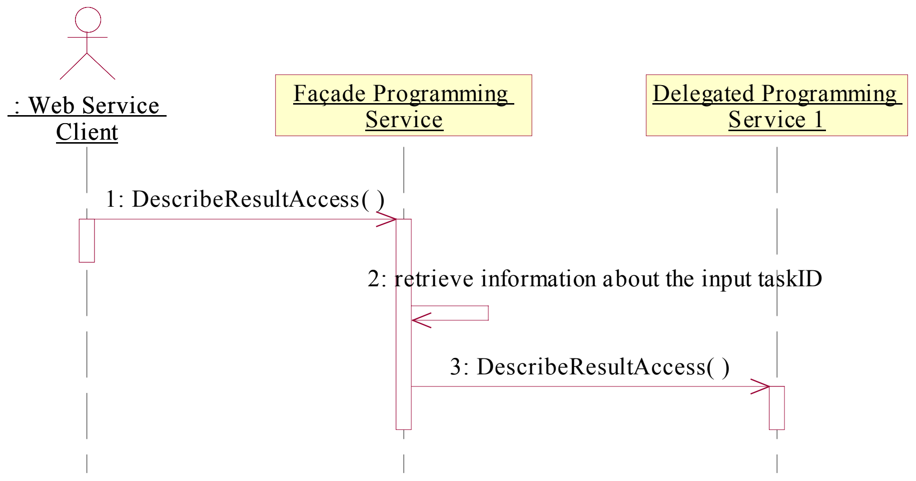

: Web Service Client
Façade Programming Service
Delegated Programming Service 1
1: DescribeResultAccess( )
2: retrieve information about the input taskID
3: DescribeResultAccess( )

Figure 20-5: DescribeResultAccess scenario.

- If the final delivery of the programming results has not been specified in the Submit response, the client has to call the DescribeAccessResult operation in order to find the address of the service allowing accessing the data acquired by the programming request.

- The Façade retrieves the data related to the taskID specified in the DescribeResultAccess request and forwards the request to the delegated instance that managed the programming request.

- The delegated programming service instance sends the reply to the facading programming service first (simplifies future request handling), which will forward the information to the client.

20.1.2 Semantic issues

None.

20.1.3 Technical issues

SOAP: Only SOAP messaging (via HTTP/POST) with document/literal style has to be used. Messages must conform to SOAP 1.2 (http://www.w3.org/TR/SOAP/). The message payload will be in the body of the SOAP envelope.

20.1.4 Other Issues

- Programming Parameters management.

  The current approach is that programming parameters are not aggregated and stored locally by the Façade Service. Instead, the DescribeGetFeasibility request as received by the client is split into parts (corresponding to the specific Delegated Programming Service instances) and forwarded accordingly to the corresponding Programming Service instances.

- Multi sensors request management.

  All Programming Service operations, apart from DescribeGetFeasibility, are single sensor and single programming request; the definition of the sensor is quite flexible instead as rather abstract sensor specifications are possible (e.g. it is possible to specify only the instrument or the satellite instead of providing the full set or input parameters). Then, the current specification is suitable for client – server interactions, but does not provides a full support for distributed multi-mission scenarios. In fact, as showed in the Implementation Guidance section, the currently supported scenario is the following: for each request, the Façade service understands the remote service to call and then forwards the request to it.

  Possible extensions that will be considered in next issues of the Programming Service specification are:

  - In case of abstract defined sensors, the GetFeasibility request can be forwarded to all remote services matching the loosely defined sensor and then all results are returned to the user as possible options to choose.

  - Support of several sensors within the same request to cover a specified area with products of different sensors. This will be analysed in the Mission Planning work that will be carried out in the frame of GMES HMA project.

Annex A
(normative)

XML schema documents

A.1. spsGetCapabilities.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:sps="http://www.opengis.net/sps/0" xmlns:ows="http://www.opengeospatial.net/ows"
xmlns:xs="http://www.w3.org/2001/XMLSchema" targetNamespace="http://www.opengis.net/sps/0"
elementFormDefault="qualified" attributeFormDefault="unqualified" xml:lang="en">
  <xs:annotation>
    <xs:appinfo>spsGetCapabilities.xsd 2005/05/11</xs:appinfo>
    <xs:documentation>
      <description>This XML Schema encodes the SPS GetCapabilities operation request and response.</description>
      <copyright>Copyright (c) 2005 Institut for Geoinformatics University of Muenster</copyright>
    </xs:documentation>
  </xs:annotation>
  <!-- ===============================================================
          imports
       =============================================================== -->
  <xs:import namespace="http://www.opengeospatial.net/ows" schemaLocation="./ows4sps.xsd"/>
  <xs:include schemaLocation="./spsContents.xsd"/>
  <!-- ===============================================================
          elements and types
       =============================================================== -->
  <xs:element name="GetCapabilities">
    <xs:annotation>
      <xs:documentation>Request to a SPS to perform the GetCapabilities operation. This operation allows a client to
retrieve service metadata (capabilities XML) providing metadata for the specific SPS server. In this XML encoding,
no "request" parameter is included, since the element name specifies the specific operation. </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="ows:GetCapabilitiesType">
          <xs:attribute name="service" type="ows:ServiceType" use="required" fixed="SPS"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <!-- ============================================================= -->
  <xs:element name="Capabilities">
    <xs:annotation>
      <xs:documentation>XML encoded SPS GetCapabilities operation response. This document provides clients with
service metadata about a specific service instance. If the server does not implement the updateSequence parameter,
the server shall always return the complete Capabilities document, without the updateSequence parameter. When the
server implements the updateSequence parameter and the GetCapabilities operation request included the
updateSequence parameter with the current value, the server shall return this element with only the "version" and
"updateSequence" attributes. Otherwise, all optional elements shall be included or not depending on the actual value
of the Sections parameter in the GetCapabilities operation request. </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="ows:CapabilitiesBaseType">
          <xs:sequence>
            <xs:element ref="sps:Contents" minOccurs="0"/>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

A.2. spsDescribeSensor.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2007 sp1 (http://www.altova.com) by Merigot (SPOT IMAGE) -->
<xs:schema xmlns:sps="http://www.opengis.net/sps/0" xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.opengis.net/sps/0" elementFormDefault="qualified"
attributeFormDefault="unqualified">
  <xs:include schemaLocation="./spsCommon.xsd"/>
  <xs:element name="DescribeSensor">
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="sps:RequestBaseType">
          <xs:sequence>
            <xs:element ref="sps:sensorID"/>
            <xs:element name="descriptionType">
              <xs:annotation>
                <xs:documentation>Indicates the type of sensor description expected : full or brief</xs:documentation>
              </xs:annotation>
              <xs:simpleType>
                <xs:restriction base="xs:string">
                  <xs:enumeration value="brief"/>
                  <xs:enumeration value="full"/>
                </xs:restriction>
              </xs:simpleType>
            </xs:element>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Future work : DescribeSensor response

A.3. spsDescribeGetFeasibility.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2007 sp1 (http://www.altova.com) by Merigot (SPOT IMAGE) -->
<xs:schema xmlns:sps="http://www.opengis.net/sps/0" xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.opengis.net/sps/0" elementFormDefault="qualified"
attributeFormDefault="unqualified">
  <xs:include schemaLocation="./spsCommon.xsd"/>
  <xs:element name="DescribeGetFeasibility">
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="sps:RequestBaseType">
          <xs:sequence>
            <xs:element ref="sps:sensorID"/>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="DescribeGetFeasibilityResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="InputParameters" type="sps:ParametersDescriptorType" minOccurs="0">
          <xs:annotation>
            <xs:documentation>Not specified if parameters are the same as DescribeSubmit
InputParameters</xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element name="OutputParameters" type="sps:ParametersDescriptorType" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

A.4. spsGetFeasibility.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2007 sp1 (http://www.altova.com) by Merigot (SPOT IMAGE) -->
<xs:schema xmlns:sps="http://www.opengis.net/sps/0" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:wns="http://www.opengis.net/wns" targetNamespace="http://www.opengis.net/sps/0"
elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:import namespace="http://www.opengis.net/wns" schemaLocation="./wns4sps.xsd"/>
  <xs:include schemaLocation="./spsCommon.xsd"/>
  <xs:element name="GetFeasibility">
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="sps:RequestBaseType">
          <xs:sequence>
            <xs:element ref="wns:NotificationTarget" minOccurs="0"/>
            <xs:element ref="sps:sensorID"/>
            <xs:element ref="sps:ID" minOccurs="0"/>
            <xs:element ref="sps:parameters" minOccurs="0"/>
            <xs:element ref="sps:TimeFrame" minOccurs="0"/>
            <xs:element name="feasibilityLevel">
              <xs:annotation>
                <xs:documentation>Possible values :
light, estimate, full</xs:documentation>
              </xs:annotation>
              <xs:simpleType>
                <xs:restriction base="xs:string">
                  <xs:enumeration value="light"/>
                  <xs:enumeration value="estimate"/>
                  <xs:enumeration value="full"/>
                </xs:restriction>
              </xs:simpleType>
            </xs:element>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="GetFeasibilityRequestAck">
    <xs:annotation>
      <xs:documentation>Acknowledgement of the GetFeasibility request</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="RequestAckStatus" type="sps:RequestAckStatusType"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="GetFeasibilityResponse">
    <xs:annotation>
      <xs:documentation>Reponse to a GetFeasibility request</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="sps:ID"/>
        <xs:element name="FeasibilityResult" minOccurs="0">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="feasibility">
                <xs:annotation>
                  <xs:documentation>describes if a request is feasible or not</xs:documentation>
                </xs:annotation>
                <xs:simpleType>
                  <xs:restriction base="xs:string">
                    <xs:enumeration value="feasible"/>
                    <xs:enumeration value="not feasible"/>
                    <xs:enumeration value="response delayed. Notification will be sent."/>
                    <xs:enumeration value="request incomplete"/>
                  </xs:restriction>
                </xs:simpleType>
              </xs:element>
              <xs:element ref="sps:Description" minOccurs="0"/>
              <xs:element ref="sps:LatestResponseTime" minOccurs="0"/>
              <xs:element name="Output" minOccurs="0">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element ref="sps:parameters"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
            <xs:attribute name="SuccessRate">
              <xs:annotation>
                <xs:documentation>Indicates the success rate for the given parameter constellation in
percent.</xs:documentation>
              </xs:annotation>
              <xs:simpleType>
                <xs:restriction base="xs:double">
                  <xs:minInclusive value="0"/>
                  <xs:maxInclusive value="100"/>
                </xs:restriction>
              </xs:simpleType>
            </xs:attribute>
          </xs:complexType>
        </xs:element>
        <xs:element ref="sps:Alternative" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="GetFeasibilityResponseAck">
    <xs:annotation>
      <xs:documentation>Acknowledgment to a GetFeasibilityResponse</xs:documentation>
    </xs:annotation>
    <xs:complexType>
```

```
      <xs:sequence>
        <xs:element name="ResponseAckStatus" type="sps:ResponseAckStatusType"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

A.5. spsDescribeSubmit.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2007 sp1 (http://www.altova.com) by Merigot (SPOT IMAGE) -->
<xs:schema xmlns:sps="http://www.opengis.net/sps/0" xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.opengis.net/sps/0" elementFormDefault="qualified"
attributeFormDefault="unqualified">
  <xs:include schemaLocation="./spsCommon.xsd"/>
  <xs:element name="DescribeSubmit">
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="sps:RequestBaseType">
          <xs:sequence>
            <xs:element ref="sps:sensorID"/>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="DescribeSubmitResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="InputParameters" type="sps:ParametersDescriptorType"/>
        <xs:element name="OutputParameters" type="sps:ParametersDescriptorType" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

A.6. spsSubmit.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2007 sp1 (http://www.altova.com) by Merigot (SPOT IMAGE) -->
<xs:schema xmlns:sps="http://www.opengis.net/sps/0" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:wns="http://www.opengis.net/wns" targetNamespace="http://www.opengis.net/sps/0"
elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:import namespace="http://www.opengis.net/wns" schemaLocation="./wns4sps.xsd"/>
  <xs:include schemaLocation="./spsCommon.xsd"/>
  <xs:element name="Submit">
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="sps:RequestBaseType">
          <xs:sequence>
            <xs:element ref="wns:NotificationTarget" minOccurs="0"/>
            <xs:choice>
              <xs:element name="sensorParam">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element ref="sps:sensorID"/>
                    <xs:element ref="sps:parameters"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element ref="sps:ID"/>
            </xs:choice>
            <xs:element ref="sps:TimeFrame" minOccurs="0"/>
            <xs:element name="statusNotification" minOccurs="0">
              <xs:annotation>
                <xs:documentation>Specifies how status notifications are sent back to the client</xs:documentation>
              </xs:annotation>
              <xs:simpleType>
                <xs:restriction base="xs:string">
                  <xs:enumeration value="once"/>
                  <xs:enumeration value="final"/>
                  <xs:enumeration value="all"/>
                </xs:restriction>
              </xs:simpleType>
            </xs:element>
            <xs:element name="DeliveryInformation" type="sps:DeliveryInformationType" minOccurs="0"/>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="SubmitRequestAck">
    <xs:annotation>
      <xs:documentation>Acknowledgement of the Submit request</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="sps:ID"/>
        <xs:element name="SubmitResult" type="sps:RequestAckStatusType"/>
        <xs:element ref="sps:Alternative" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="SubmitResponse">
    <xs:annotation>
      <xs:documentation>Asynchronous response to a Submit request</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="ProgressReport" type="sps:ProgressReportType"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="SubmitResponseAck">
    <xs:annotation>
      <xs:documentation>Acknowledgment to a SubmitResponse</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="ResponseAckStatus" type="sps:ResponseAckStatusType"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

A.7. spsGetStatus.xsd

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

```
<!-- edited with XMLSpy v2007 sp1 (http://www.altova.com) by Merigot (SPOT IMAGE) -->
<xs:schema xmlns:sps="http://www.opengis.net/sps/0" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:wns="http://www.opengis.net/wns" targetNamespace="http://www.opengis.net/sps/0"
elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:import namespace="http://www.opengis.net/wns" schemaLocation="./wns4sps.xsd"/>
  <xs:include schemaLocation="./spsCommon.xsd"/>
  <xs:element name="GetStatus">
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="sps:RequestBaseType">
          <xs:sequence>
            <xs:element ref="wns:NotificationTarget" minOccurs="0"/>
            <xs:element ref="sps:ID"/>
            <xs:element name="DateFrom" type="xs:dateTime" minOccurs="0"/>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="GetStatusResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="ProgressReport" type="sps:ProgressReportType"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

A.8. spsCancel.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:sps="http://www.opengis.net/sps/0" xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.opengis.net/sps/0" elementFormDefault="qualified"
attributeFormDefault="unqualified">
  <xs:include schemaLocation="./spsCommon.xsd"/>
  <xs:element name="Cancel">
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="sps:RequestBaseType">
          <xs:sequence>
            <xs:element ref="sps:ID"/>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="CancelResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="sps:ID"/>
        <xs:element name="status">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="confirmed"/>
              <xs:enumeration value="rejected"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
        <xs:element ref="sps:Description" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

A.9. spsUpdate.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:sps="http://www.opengis.net/sps/0" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:wns="http://www.opengis.net/wns" targetNamespace="http://www.opengis.net/sps/0"
elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:import namespace="http://www.opengis.net/wns" schemaLocation="./wns4sps.xsd"/>
  <xs:include schemaLocation="./spsCommon.xsd"/>
  <xs:element name="Update">
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="sps:RequestBaseType">
          <xs:sequence>
            <xs:element ref="sps:ID"/>
            <xs:element ref="wns:NotificationTarget" minOccurs="0"/>
            <xs:element ref="sps:parameters" minOccurs="0"/>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="UpdateResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="sps:ID"/>
        <xs:element name="status">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="confirmed"/>
              <xs:enumeration value="rejected"/>
              <xs:enumeration value="request incomplete"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
        <xs:element ref="sps:Description" minOccurs="0"/>
        <xs:element ref="sps:EstimatedToC" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

A.10. spsDescribeResultAccess.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2007 sp1 (http://www.altova.com) by Merigot (SPOT IMAGE) -->
<xs:schema xmlns:sps="http://www.opengis.net/sps/0" xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.opengis.net/sps/0" elementFormDefault="qualified"
attributeFormDefault="unqualified">
  <xs:include schemaLocation="./spsCommon.xsd"/>
  <xs:element name="DescribeResultAccess">
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base="sps:RequestBaseType">
          <xs:choice>
```

```
            <xs:sequence>
              <xs:element ref="sps:ID"/>
              <xs:element name="DateFrom" type="xs:dateTime" minOccurs="0"/>
            </xs:sequence>
            <xs:element ref="sps:sensorID"/>
          </xs:choice>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="DescribeResultAccessResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="service" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="ServiceType" type="xs:string"/>
              <xs:element name="ServiceURL" type="xs:anyURI"/>
              <xs:element name="ServiceRequest" minOccurs="0">
                <xs:annotation>
                  <xs:documentation>If a POST request has to be used to get the data from the service, the request can be
incorporated in this element.</xs:documentation>
                </xs:annotation>
                <xs:complexType>
                  <xs:sequence>
                    <xs:any namespace="##any"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element name="productId" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType/>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

A.11. spsCommon.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2007 sp1 (http://www.altova.com) by Merigot (SPOT IMAGE) -->
<!-- edited with XMLSPY v2004 rel. 2 U (http://www.xmlspy.com) by Institut für Geoinformatik (Institut für
Geoinformatik) -->
<xs:schema xmlns:swe="http://www.opengis.net/swe/0" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:sps="http://www.opengis.net/sps/0" xmlns:wns="http://www.opengis.net/wns"
targetNamespace="http://www.opengis.net/sps/0" elementFormDefault="qualified"
attributeFormDefault="unqualified" xml:lang="en">
  <!-- ===================================================================
            includes and imports
       =================================================================== -->
  <xs:import namespace="http://www.opengis.net/wns" schemaLocation="./wns4sps.xsd"/>
  <xs:import namespace="http://www.opengis.net/swe/0" schemaLocation="./swe4sps.xsd"/>
  <!-- ===================================================================
            elements and types
       =================================================================== -->
  <xs:complexType name="RequestBaseType">
```

```
    <xs:annotation>
      <xs:documentation>XML encoded SPS operation request base, for all operations except Get Capabilities. In this
XML encoding, no "request" parameter is included, since the element name specifies the specific operation.
</xs:documentation>
    </xs:annotation>
    <xs:attribute name="service" type="xs:string" use="required" fixed="SPS">
      <xs:annotation>
        <xs:documentation>Service type identifier. </xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="version" type="xs:string" use="required" fixed="1.0.0">
      <xs:annotation>
        <xs:documentation>Specification version for SPS version and operation.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
  </xs:complexType>
  <xs:complexType name="ParameterDescriptorType">
    <xs:sequence>
      <xs:element ref="sps:Description" minOccurs="0"/>
      <xs:element name="definition" maxOccurs="unbounded">
        <xs:annotation>
          <xs:documentation>implicit OR : a parameter may be described by more than one
definition</xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:choice>
            <xs:element name="commonData">
              <xs:complexType>
                <xs:sequence>
                  <xs:group ref="swe:AnyData"/>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
            <xs:element name="TaskMessageDefinition">
              <xs:annotation>
                <xs:documentation>links to a URI dictionary whrere the taskMessage is defined
properly.</xs:documentation>
              </xs:annotation>
              <xs:complexType>
                <xs:simpleContent>
                  <xs:extension base="xs:anyURI">
                    <xs:attribute name="definition" type="swe:definitionReference"/>
                  </xs:extension>
                </xs:simpleContent>
              </xs:complexType>
            </xs:element>
            <xs:element name="GeometryDefinition">
              <xs:annotation>
                <xs:documentation>enumerates gml:Point, gml:Line, gml:Polygon as possible values</xs:documentation>
              </xs:annotation>
              <xs:complexType>
                <xs:simpleContent>
                  <xs:extension base="xs:QName">
                    <xs:attribute name="definition" type="swe:definitionReference"/>
                  </xs:extension>
                </xs:simpleContent>
              </xs:complexType>
            </xs:element>
            <xs:element ref="sps:ParameterDescriptor" maxOccurs="unbounded">
              <xs:annotation>
                <xs:documentation>A parameter may contain a list of parameters, and so on</xs:documentation>
              </xs:annotation>
            </xs:element>
          </xs:choice>
        </xs:complexType>
      </xs:element>
      <xs:element name="restriction" minOccurs="0">
        <xs:annotation>
          <xs:documentation>optional. Only used if the client has to choose one or many of the provided values.
</xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element ref="sps:Parameter"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="cardinality" minOccurs="0">
        <xs:annotation>
          <xs:documentation>Defines the number of input objects that could be provided.</xs:documentation>
        </xs:annotation>
        <xs:simpleType>
          <xs:union>
            <xs:simpleType>
              <xs:restriction base="xs:int">
                <xs:minExclusive value="0"/>
              </xs:restriction>
            </xs:simpleType>
            <xs:simpleType>
              <xs:restriction base="xs:string">
                <xs:enumeration value="unbounded"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:union>
        </xs:simpleType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="parameterID" type="xs:token" use="required"/>
    <xs:attribute name="use" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="required"/>
          <xs:enumeration value="optional"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="updateable" type="xs:boolean" use="required"/>
</xs:complexType>
<xs:complexType name="ParametersDescriptorType">
    <xs:sequence>
      <xs:element ref="sps:Description" minOccurs="0"/>
      <xs:element ref="sps:ParameterDescriptor" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="ProgressReportType">
    <xs:sequence>
      <xs:element ref="sps:ID"/>
      <xs:element name="status">
        <xs:annotation>
          <xs:documentation>defines if the request is being processed, rejected or failed to process due to insuffenent
parametrization.</xs:documentation>
        </xs:annotation>
        <xs:simpleType>
          <xs:restriction base="xs:string">
```

```
            <xs:enumeration value="unknown"/>
            <xs:enumeration value="in operation"/>
            <xs:enumeration value="finished"/>
            <xs:enumeration value="not yet started"/>
            <xs:enumeration value="cancelled"/>
            <xs:enumeration value="delayed"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element ref="sps:Description" minOccurs="0"/>
      <xs:element ref="sps:EstimatedToC" minOccurs="0"/>
      <xs:element name="Output" minOccurs="0">
        <xs:complexType>
          <xs:sequence>
            <xs:element ref="sps:Description" minOccurs="0"/>
            <xs:element ref="sps:parameters"/>
            <xs:element name="DateFrom" type="xs:dateTime" minOccurs="0"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="RequestAckStatusType">
    <xs:sequence>
      <xs:element name="status">
        <xs:annotation>
          <xs:documentation>defines if the request is being processed, rejected or failed to process due to insuffencent
parametrization.</xs:documentation>
        </xs:annotation>
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="confirmed"/>
            <xs:enumeration value="rejected"/>
            <xs:enumeration value="pending"/>
            <xs:enumeration value="request incomplete"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element ref="sps:Description" minOccurs="0"/>
      <xs:element ref="sps:LatestResponseTime" minOccurs="0"/>
      <xs:element ref="sps:EstimatedToC" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
  <xs:simpleType name="ResponseAckStatusType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Ok"/>
    </xs:restriction>
  </xs:simpleType>
  <!--ID elements-->
  <xs:element name="sensorID" type="xs:token">
    <xs:annotation>
      <xs:documentation>Identifies a sensor. Unique to every SPS instance.</xs:documentation>
    </xs:annotation>
  </xs:element>
  <xs:element name="ID" type="xs:token">
    <xs:annotation>
      <xs:documentation>Unique ID which either references a feasibility study, a reservation or a task - possible
meanings are depending on the actual operation.</xs:documentation>
    </xs:annotation>
  </xs:element>
  <!--elements used for defining task input-->
  <xs:element name="Parameter">
```

```
    <xs:annotation>
      <xs:documentation>Contains the input for a certain tasking parameter.</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="value" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:any processContents="skip"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="parameterID" type="xs:token" use="required"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="ParameterDescriptor" type="sps:ParameterDescriptorType">
    <xs:annotation>
      <xs:documentation>Defines the input required to task a sensor.</xs:documentation>
    </xs:annotation>
  </xs:element>
  <!--additional elements-->
  <xs:element name="parameters">
    <xs:annotation>
      <xs:documentation>Contains a list of parameters.</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="sps:Parameter" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="TimeFrame" type="xs:dateTime">
    <xs:annotation>
      <xs:documentation>Maximum point in time a request keeps being valid. </xs:documentation>
    </xs:annotation>
  </xs:element>
  <xs:element name="Alternative">
    <xs:annotation>
      <xs:documentation>Provides an alternative parameter constellation.</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="sps:Description" minOccurs="0"/>
        <xs:element ref="sps:parameters"/>
      </xs:sequence>
      <xs:attribute name="SuccessRate">
        <xs:annotation>
          <xs:documentation>Indicates the success rate for the given alternative in percent. If omitted the success rate
equals 100%.</xs:documentation>
        </xs:annotation>
        <xs:simpleType>
          <xs:restriction base="xs:double">
            <xs:maxInclusive value="100"/>
            <xs:minExclusive value="0"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
  </xs:element>
  <xs:element name="LatestResponseTime" type="xs:dateTime">
    <xs:annotation>
```

```
      <xs:documentation>Denotes the point in time when the notification that contains the definite response to a
delayed operation request will be sent. If no notification has been received until that time the operation shall be
deemed to have been rejected or failed.</xs:documentation>
    </xs:annotation>
  </xs:element>
  <xs:element name="EstimatedToC" type="xs:dateTime">
    <xs:annotation>
      <xs:documentation>Estimated Time of Completion gives a hint when the task might be
completed.</xs:documentation>
    </xs:annotation>
  </xs:element>
  <xs:element name="Description" type="xs:string">
    <xs:annotation>
      <xs:documentation>Contains a simple description.</xs:documentation>
    </xs:annotation>
  </xs:element>
</xs:schema>
```

Annex B
(informative)

Example of XML documents

B.1. Example of DescribeSubmit response

```
<?xml version="1.0" encoding="UTF-8"?>
<DescribeSubmitResponse xmlns="http://www.opengis.net/sps/0" xmlns:gml="http://www.opengis.net/gml"
xmlns:swe="http://www.opengis.net/swe/0" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.opengis.net/sps/0
D:\users\pmerigot\projets\OGC\dev\siows\doc\sps\schemas\sweCommon_current\swe\sps\current_pm\spsDescribeSu
bmit.xsd">
  <InputParameters>
    <Description>List of parameters a client has to provide to task SPOT5 sensor</Description>
    <ParameterDescriptor use="required" parameterID="sensorID" updateable="false">
      <Description>Sensor identifier</Description>
      <definition>
        <commonData>
          <swe:Text/>
        </commonData>
      </definition>
    </ParameterDescriptor>
    <ParameterDescriptor use="required" parameterID="priorityLevel" updateable="false">
      <Description>Programming Priority level</Description>
      <definition>
        <commonData>
          <swe:Category>
            <swe:constraint>
              <swe:AllowedTokens>
                <swe:tokenList>Routine</swe:tokenList>
                <swe:tokenList>Urgent</swe:tokenList>
                <swe:tokenList>Crisis</swe:tokenList>
              </swe:AllowedTokens>
            </swe:constraint>
          </swe:Category>
        </commonData>
      </definition>
    </ParameterDescriptor>
    <ParameterDescriptor use="required" parameterID="AcquisitionParameters" updateable="false">
      <definition>
        <ParameterDescriptor use="required" parameterID="Resolution" updateable="false">
          <definition>
            <commonData>
              <swe:Quantity>
                <swe:uom xlink:href="urn:ogc:def:unit:meter"/>
                <swe:constraint>
                  <swe:AllowedValues>
                    <swe:valueList>2.5 5 10</swe:valueList>
                  </swe:AllowedValues>
                </swe:constraint>
              </swe:Quantity>
            </commonData>
          </definition>
        </ParameterDescriptor>
        <ParameterDescriptor use="required" parameterID="AcquisitionMode" updateable="false">
          <definition>
  <commonData>
   <swe:Category>
    <swe:constraint>
     <swe:AllowedTokens>
      <swe:tokenList>COLOR</swe:tokenList>
      <swe:tokenList>BLACK AND WHITE</swe:tokenList>
     </swe:AllowedTokens>
    </swe:constraint>
   </swe:Category>
  </commonData>
 </definition>
</ParameterDescriptor>
<ParameterDescriptor use="required" parameterID="ProgrammingRequestMode" updateable="false">
 <Description>Programming type</Description>
 <definition>
  <commonData>
   <swe:Category>
    <swe:constraint>
     <swe:AllowedTokens>
      <swe:tokenList>SCENE</swe:tokenList>
      <swe:tokenList>MONOPASS COVERAGE</swe:tokenList>
      <swe:tokenList>MULTIPASS COVERAGE</swe:tokenList>
     </swe:AllowedTokens>
    </swe:constraint>
   </swe:Category>
  </commonData>
 </definition>
</ParameterDescriptor>
<ParameterDescriptor use="required" parameterID="SurveyPeriods" updateable="false">
 <definition>
  <ParameterDescriptor use="required" parameterID="Scene" updateable="false">
   <definition>
    <ParameterDescriptor use="required" parameterID="SurveyPeriod" updateable="false">
     <Description>Start date and end date of acquisition</Description>
     <definition>
      <commonData>
       <swe:TimeRange/>
      </commonData>
     </definition>
    </ParameterDescriptor>
   </definition>
   <definition>
    <ParameterDescriptor use="required" parameterID="FrameNumber" updateable="false">
     <definition>
      <commonData>
       <swe:Count/>
      </commonData>
     </definition>
    </ParameterDescriptor>
   </definition>
   <definition>
    <ParameterDescriptor use="required" parameterID="OrbitalParameters" updateable="false">
     <definition>
      <ParameterDescriptor use="required" parameterID="OrbitNumber" updateable="false">
       <Description>Orbit number</Description>
       <definition>
        <commonData>
         <swe:Count/>
        </commonData>
       </definition>
      </ParameterDescriptor>
      <ParameterDescriptor use="required" parameterID="OrbitDirection" updateable="false">
```

```
                  <Description>Orbit direction</Description>
                  <definition>
                    <commonData>
                      <swe:Category>
                        <swe:constraint>
                          <swe:AllowedTokens>
                            <swe:tokenList>ASCENDING</swe:tokenList>
                            <swe:tokenList>DESCENDING</swe:tokenList>
                          </swe:AllowedTokens>
                        </swe:constraint>
                      </swe:Category>
                    </commonData>
                  </definition>
                </ParameterDescriptor>
                <ParameterDescriptor use="required" parameterID="ANX_StartTime" updateable="false">
                  <Description>Time in millisecond of the acquisition start with respect the ascending node
crossing</Description>
                  <definition>
                    <commonData>
                      <swe:Count>
                        <swe:uom xlink:href="urn:ogc:def:unit:millisecond"></swe:uom>
                      </swe:Count>
                    </commonData>
                  </definition>
                </ParameterDescriptor>
                <ParameterDescriptor use="required" parameterID="ANX_StopTime" updateable="false">
                  <Description>Time in millisecond of the acquisition end with respect the ascending node
crossing</Description>
                  <definition>
                    <commonData>
                      <swe:Count>
                        <swe:uom xlink:href="urn:ogc:def:unit:millisecond"/>
                      </swe:Count>
                    </commonData>
                  </definition>
                </ParameterDescriptor>
              </definition>
            </ParameterDescriptor>
          </definition>
        </ParameterDescriptor>
      </definition>
      <definition>
        <ParameterDescriptor use="required" parameterID="SingleCoverage" updateable="false">
          <definition>
            <ParameterDescriptor use="required" parameterID="SurveyPeriod" updateable="false">
              <definition>
                <commonData>
                  <swe:TimeRange/>
                </commonData>
              </definition>
            </ParameterDescriptor>
          </definition>
          <definition>
            <ParameterDescriptor use="required" parameterID="SeveralCoverage"
updateable="false"></ParameterDescriptor>
          </definition>
        </ParameterDescriptor>
      </definition>
    </ParameterDescriptor>
    </definition>
    </ParameterDescriptor>
    <ParameterDescriptor use="required" parameterID="GeometricCoverageCharacteristics" updateable="false">
```

```
     <definition>
      <ParameterDescriptor use="required" parameterID="Mono" updateable="false">
       <Description>Mono acquisition</Description>
       <definition>
        <ParameterDescriptor use="required" parameterID="IncidenceAngle" updateable="false">
         <definition>
          <ParameterDescriptor use="required" parameterID="AngleMin" updateable="false">
           <Description>Minimum incidence angle</Description>
           <definition>
            <commonData>
             <swe:QuantityRange>
              <swe:uom xlink:href="urn:ogc:def:uom:OGC:1.0:degree"></swe:uom>
             </swe:QuantityRange>
            </commonData>
           </definition>
          </ParameterDescriptor>
          <ParameterDescriptor use="required" parameterID="AngleMax" updateable="false">
           <Description>Maximum incidence angle</Description>
           <definition>
            <commonData>
             <swe:QuantityRange>
              <swe:uom xlink:href="urn:ogc:def:uom:OGC:1.0:degree"></swe:uom>
             </swe:QuantityRange>
            </commonData>
           </definition>
          </ParameterDescriptor>
         </definition>
        </ParameterDescriptor>
       </definition>
      </ParameterDescriptor>
     </definition>
     <definition>
      <ParameterDescriptor use="required" parameterID="Stereo" updateable="false">
       <Description>Stereo acquisition</Description>
       <definition>
        <ParameterDescriptor use="required" parameterID="IncidenceAngle1"
updateable="false"></ParameterDescriptor>
        <ParameterDescriptor use="required" parameterID="IncidenceAngle2"
updateable="false"></ParameterDescriptor>
        <ParameterDescriptor use="required" parameterID="Constraints" updateable="false">
         <definition>
          <ParameterDescriptor use="required" parameterID="BHMin"
updateable="false"></ParameterDescriptor>
          <ParameterDescriptor use="required" parameterID="BHMax"
updateable="false"></ParameterDescriptor>
          <ParameterDescriptor use="required" parameterID="MaxCoupleDelay"
updateable="false"></ParameterDescriptor>
         </definition>
        </ParameterDescriptor>
       </definition>
      </ParameterDescriptor>
     </definition>
    </ParameterDescriptor>
    <ParameterDescriptor use="required" parameterID="RegionOfInterest" updateable="false">
            <definition>
                <GeometryDefinition/>
            </definition>
    </ParameterDescriptor>
    <ParameterDescriptor use="required" parameterID="ValidationParameters" updateable="false">
     <definition>
      <ParameterDescriptor parameterID="cloudCoverPercentage" use="optional" updateable="false">
       <Description>Maximum allowed cloud coverage</Description>
```

```
          <definition>
            <commonData>
              <swe:Count>
                <swe:uom xlink:href="urn:ogc:def:unit:percentage"/>
                <swe:constraint>
                  <swe:AllowedValues>
                    <swe:min>0</swe:min>
                    <swe:max>100</swe:max>
                  </swe:AllowedValues>
                </swe:constraint>
              </swe:Count>
            </commonData>
          </definition>
        </ParameterDescriptor>
        <ParameterDescriptor parameterID="snowCoverPercentage" use="optional" updateable="false">
          <Description>Maximum allowed snow coverage</Description>
          <definition>
            <commonData>
              <swe:Count>
                <swe:uom xlink:href="urn:ogc:def:unit:percentage"/>
                <swe:constraint>
                  <swe:AllowedValues>
                    <swe:min>0</swe:min>
                    <swe:max>100</swe:max>
                  </swe:AllowedValues>
                </swe:constraint>
              </swe:Count>
            </commonData>
          </definition>
        </ParameterDescriptor>
        <ParameterDescriptor parameterID="hazeAccepted" use="optional" updateable="false">
          <Description/>
          <definition>
            <commonData>
              <swe:Count>
                <swe:uom xlink:href="urn:ogc:def:unit:percentage"/>
                <swe:constraint>
                  <swe:AllowedValues>
                    <swe:min>0</swe:min>
                    <swe:max>100</swe:max>
                  </swe:AllowedValues>
                </swe:constraint>
              </swe:Count>
            </commonData>
          </definition>
        </ParameterDescriptor>
      </definition>
    </ParameterDescriptor>
  </InputParameters>
</DescribeSubmitResponse>
```

B.2. Examples of DescribeSensor response

The DescribeSensor operation may return either a complete or a brief description of the sensors (cf. § 11).

B.2.1. Example of brief description (SPOT 10 meter 4 bands):

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- Author: Alexandre Robin - Sensia Software LLC        -->
<!-- Creation Date: 2007-02-02                            -->
<!-- Copyright: SpotImage S.A., France                    -->
<sml:SensorML xmlns:sml="http://www.opengis.net/sensorML/1.0" xmlns:swe="http://www.opengis.net/swe/1.0"
xmlns:gml="http://www.opengis.net/gml" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xlink="http://www.w3.org/1999/xlink" xsi:schemaLocation="http://www.opengis.net/sensorML/1.0
../Schema/sensorML/sensorML.xsd" version="1.0">
  <!-- -->
  <sml:member xlink:role="urn:x-ogc:def:dictionary:ESA:documentRoles:v01#system_summary">
    <!-- -->
    <sml:System gml:id="SPOT_10M_COLOR">
      <!-- ================================================== -->
      <!--             System Description              -->
      <!-- ================================================== -->
      <gml:description> Configuration of the SPOT constellation for 10m Color Imagery </gml:description>
      <!-- ================================================== -->
      <!--             System Identifiers              -->
      <!-- ================================================== -->
      <sml:identification>
        <sml:IdentifierList>
          <sml:identifier name="System UID">
            <sml:Term definition="urn:x-ogc:def:identifier:OGC:uuid">
              <sml:value>urn:x-ogc:object:platform:ESA:SPOT:10mColor:v01</sml:value>
            </sml:Term>
          </sml:identifier>
          <sml:identifier name="Short Name">
            <sml:Term definition="urn:x-ogc:def:identifier:OGC:shortName">
              <sml:value>SPOT 10m Color</sml:value>
            </sml:Term>
          </sml:identifier>
        </sml:IdentifierList>
      </sml:identification>
      <!-- ================================================== -->
      <!--            System Capabilities              -->
      <!-- ================================================== -->
      <sml:capabilities>
        <swe:DataRecord>
          <swe:field name="Band Type">
            <swe:Category  definition="urn:x-ogc:def:classifier:OGC:bandType">
              <swe:codeSpace xlink:href="urn:x-ogc:def:dictionary:ESA:bandTypes:v01"/>
              <swe:value>COLOR</swe:value>
            </swe:Category>
          </swe:field>
          <swe:field name="Ground Resolution">
            <swe:Quantity definition="urn:x-ogc:def:phenomenon:ESA:groundResolution">
              <swe:uom code="m"/>
              <swe:value>10</swe:value>
            </swe:Quantity>
          </swe:field>
          <swe:field name="Latitude Coverage">
            <swe:QuantityRange definition="urn:x-ogc:def:phenomenon:OGC:latitude">
              <swe:uom code="deg"/>
              <swe:value>-85 +85</swe:value>
            </swe:QuantityRange>
          </swe:field>
          <swe:field name="Longitude Coverage">
            <swe:QuantityRange definition="urn:x-ogc:def:phenomenon:OGC:longitude">
              <swe:uom code="deg"/>
              <swe:value>-180 +180</swe:value>
            </swe:QuantityRange>
          </swe:field>
        </swe:DataRecord>
      </sml:capabilities>
      <!-- ================================================== -->
      <!--            System Components            -->
      <!-- ================================================== -->
      <sml:components>
        <sml:ComponentList>
          <sml:component name="SPOT4-HRVIR1">
            <sml:System>
              <sml:identification>
                <sml:IdentifierList>
                  <sml:identifier name="System UID">
                    <sml:Term definition="urn:x-ogc:def:identifier:OGC:uuid">
                      <sml:value>urn:x-ogc:object:instrument:ESA:SPOT4:HRVIR1:v01</sml:value>
                    </sml:Term>
                  </sml:identifier>
                  <sml:identifier name="Platform UID">
                    <sml:Term definition="urn:x-ogc:def:identifier:OGC:uuid">
                      <sml:value>urn:x-ogc:object:instrument:ESA:SPOT4:v01</sml:value>
                    </sml:Term>
                  </sml:identifier>
                  <sml:identifier name="Short Name">
                    <sml:Term definition="urn:x-ogc:def:identifier:OGC:shortName">
                      <sml:value>SPOT4 HRVIR1</sml:value>
                    </sml:Term>
                  </sml:identifier>
                </sml:IdentifierList>
              </sml:identification>
            </sml:System>
          </sml:component>
          <sml:component name="SPOT4-HRVIR2">
            <sml:System>
              <sml:identification>
                <sml:IdentifierList>
                  <sml:identifier name="System UID">
                    <sml:Term definition="urn:x-ogc:def:identifier:OGC:uuid">
                      <sml:value>urn:x-ogc:object:instrument:ESA:SPOT4:HRVIR2:v01</sml:value>
                    </sml:Term>
                  </sml:identifier>
                  <sml:identifier name="Platform UID">
                    <sml:Term definition="urn:x-ogc:def:identifier:OGC:uuid">
                      <sml:value>urn:x-ogc:object:instrument:ESA:SPOT4:v01</sml:value>
                    </sml:Term>
                  </sml:identifier>
                  <sml:identifier name="Short Name">
                    <sml:Term definition="urn:x-ogc:def:identifier:OGC:shortName">
                      <sml:value>SPOT4 HRVIR2</sml:value>
                    </sml:Term>
                  </sml:identifier>
                </sml:IdentifierList>
              </sml:identification>
            </sml:System>
          </sml:component>
```

```
          <sml:component name="SPOT5-HRG1">
            <sml:System>
              <sml:identification>
                <sml:IdentifierList>
                  <sml:identifier name="System UID">
                    <sml:Term definition="urn:x-ogc:def:identifier:OGC:uuid">
                      <sml:value>urn:x-ogc:object:instrument:ESA:SPOT5:HRG1:v01</sml:value>
                    </sml:Term>
                  </sml:identifier>
                  <sml:identifier name="Platform UID">
                    <sml:Term definition="urn:x-ogc:def:identifier:OGC:uuid">
                      <sml:value>urn:x-ogc:object:instrument:ESA:SPOT5:v01</sml:value>
                    </sml:Term>
                  </sml:identifier>
                  <sml:identifier name="Short Name">
                    <sml:Term definition="urn:x-ogc:def:identifier:OGC:shortName">
                      <sml:value>SPOT5 HRG1</sml:value>
                    </sml:Term>
                  </sml:identifier>
                </sml:IdentifierList>
              </sml:identification>
            </sml:System>
          </sml:component>
          <sml:component name="SPOT5-HRG2">
            <sml:System>
              <sml:identification>
                <sml:IdentifierList>
                  <sml:identifier name="System UID">
                    <sml:Term definition="urn:x-ogc:def:identifier:OGC:uuid">
                      <sml:value>urn:x-ogc:object:instrument:ESA:SPOT5:HRG2:v01</sml:value>
                    </sml:Term>
                  </sml:identifier>
                  <sml:identifier name="Platform UID">
                    <sml:Term definition="urn:x-ogc:def:identifier:OGC:uuid">
                      <sml:value>urn:x-ogc:object:instrument:ESA:SPOT5:v01</sml:value>
                    </sml:Term>
                  </sml:identifier>
                  <sml:identifier name="Short Name">
                    <sml:Term definition="urn:x-ogc:def:identifier:OGC:shortName">
                      <sml:value>SPOT5 HRG2</sml:value>
                    </sml:Term>
                  </sml:identifier>
                </sml:IdentifierList>
              </sml:identification>
            </sml:System>
          </sml:component>
        </sml:ComponentList>
      </sml:components>
    </sml:System>
  </sml:member>
</sml:SensorML>
```

B.2.2. Example of complete description (SPOT 5 HRS):

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Author: Alexandre Robin - Sensia Software LLC        -->
<!-- Creation Date: 2006-01-30                            -->
<!-- Copyright: SpotImage S.A., France                    -->
<sml:SensorML
  xmlns:sml="http://www.opengis.net/sensorML/1.0"
  xmlns:swe="http://www.opengis.net/swe/1.0"
  xmlns:gml="http://www.opengis.net/gml"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xlink="http://www.w3.org/1999/xlink"
     xsi:schemaLocation="http://www.opengis.net/sensorML/1.0 ../Schema/sensorML/sensorML.xsd"
version="1.0">
  <!-- -->
  <sml:member xlink:role="urn:x-ogc:def:dictionary:ESA:documentRoles:v01#instrument_capabilities">
    <!-- -->
    <sml:System gml:id="SPOT5_HRS">
      <!-- ================================================= -->
      <!--                System Description                 -->
      <!-- ================================================= -->
      <gml:description>
        The HRS instrument on board SPOT5 is a high resolution stereoscopic imager. It uses a pushbroom scanning
        technique and two different ccd detectors - one pointing forward and one pointing backward - to acquire
        stereoscopic panchromatic images of the earth with 10m resolution.
      </gml:description>
      <!-- ================================================= -->
      <!--                System Identifiers                 -->
      <!-- ================================================= -->
      <sml:identification>
        <sml:IdentifierList>
          <sml:identifier name="System UID">
            <sml:Term definition="urn:x-ogc:def:identifier:OGC:uuid">
              <sml:value>urn:x-ogc:object:instrument:ESA:SPOT5:HRS:v01</sml:value>
            </sml:Term>
          </sml:identifier>
          <sml:identifier name="Short Name">
            <sml:Term definition="urn:x-ogc:def:identifier:OGC:shortName">
              <sml:value>SPOT5 HRS</sml:value>
            </sml:Term>
          </sml:identifier>
          <sml:identifier name="Long Name">
            <sml:Term definition="urn:x-ogc:def:identifier:OGC:longName">
              <sml:value>Spot5 High Resolution Stereoscopic</sml:value>
            </sml:Term>
          </sml:identifier>
        </sml:IdentifierList>
      </sml:identification>
      <!-- ================================================= -->
      <!--                System Classifiers                 -->
      <!-- ================================================= -->
      <sml:classification>
        <sml:ClassifierList>
          <sml:classifier name="Instrument Type">
            <sml:Term definition="urn:x-ogc:def:classifier:OGC:sensorType">
              <sml:codeSpace xlink:href="urn:x-ogc:def:dictionary:ESA:instrumentTypes:v01"/>
              <sml:value>Stereo Imaging Radiometer</sml:value>
            </sml:Term>
          </sml:classifier>
          <sml:classifier name="Acquisition Method">
            <sml:Term definition="urn:x-ogc:def:classifier:OGC:sensorType">
              <sml:codeSpace xlink:href="urn:x-ogc:def:dictionary:ESA:acquisitionMethods:v01"/>
              <sml:value>Pushbroom</sml:value>
            </sml:Term>
          </sml:classifier>
          <sml:classifier name="Application">
            <sml:Term definition="urn:x-ogc:def:classifier:OGC:application">
              <sml:codeSpace xlink:href="urn:x-ogc:def:dictionary:ESA:instrumentApplications:v01"/>
              <sml:value>Land - Topography</sml:value>
            </sml:Term>
          </sml:classifier>
        </sml:ClassifierList>
      </sml:classification>
      <!-- ================================================= -->
      <!--      Temporal Validity of this description      -->
      <!-- ================================================= -->
      <sml:validTime>
        <gml:TimePeriod>
          <gml:beginPosition>2002-05-04T00:00:00Z</gml:beginPosition>
          <gml:endPosition indeterminatePosition="now"/>
        </gml:TimePeriod>
      </sml:validTime>
      <!-- ================================================= -->
      <!--      Instrument Geometric Characteristics       -->
      <!-- ================================================= -->
      <sml:characteristics>
        <swe:DataRecord>
          <gml:name>Geometric Characteristics</gml:name>
          <swe:field name="Across-Track FOV">
            <swe:Quantity definition="urn:x-ogc:def:phenomenon:ESA:acrossTrackFov">
              <gml:description>Instrument field of view at nadir</gml:description>
              <swe:uom code="deg"/>
              <swe:value>8</swe:value>
            </swe:Quantity>
          </swe:field>
          <swe:field name="Swath Width">
            <swe:Quantity definition="urn:x-ogc:def:phenomenon:ESA:nadirSwathWidth">
              <gml:description>Nominal swath width at nadir</gml:description>
              <swe:uom code="km"/>
              <swe:value>120</swe:value>
            </swe:Quantity>
          </swe:field>
          <swe:field name="Ground Location Accuracy">
            <swe:Quantity definition="urn:x-ogc:def:phenomenon:ESA:groundLocationAccuracy">
              <gml:description>Positioning accuracy of the acquired images on the ground</gml:description>
              <swe:uom code="m"/>
              <swe:value>50</swe:value>
            </swe:Quantity>
          </swe:field>
        </swe:DataRecord>
      </sml:characteristics>
      <!-- ================================================= -->
      <!--     Instrument Measurement Characteristics      -->
      <!-- ================================================= -->
      <sml:characteristics>
        <swe:DataRecord>
          <gml:name>Measurement Characteristics</gml:name>
          <swe:field name="Instrument Mode">
            <swe:Category definition="urn:x-ogc:def:identifier:ESA:instrumentMode">
              <gml:description>Mass of the instrument</gml:description>
              <swe:value>STEREO</swe:value>
            </swe:Category>
          </swe:field>
```

```
          <swe:field name="Number of Bands">
            <swe:Count definition="urn:x-ogc:def:data:ESA:numberOfBands">
              <gml:description>Number of bands for this instrument configuration</gml:description>
              <swe:value>1</swe:value>
            </swe:Count>
          </swe:field>
        </swe:DataRecord>
      </sml:characteristics>
      <!-- ================================================= -->
      <!--       Instrument Physical Characteristics         -->
      <!-- ================================================= -->
      <sml:characteristics>
        <swe:DataRecord>
          <gml:name>Physical Characteristics</gml:name>
          <swe:field name="Mass">
            <swe:Quantity definition="urn:x-ogc:def:phenomenon:OGC:mass">
              <swe:uom code="kg"/>
              <swe:value>356</swe:value>
            </swe:Quantity>
          </swe:field>
          <swe:field name="Maximum Power Consumption">
            <swe:Quantity definition="urn:x-ogc:def:phenomenon:ESA:maximumPowerConsumption">
              <gml:description>Maximum electrical power consumed by the instrument in any
mode</gml:description>
              <swe:uom code="W"/>
              <swe:value>344</swe:value>
            </swe:Quantity>
          </swe:field>
        </swe:DataRecord>
      </sml:characteristics>
      <!-- ================================================= -->
      <!--        Instrument Pointing Capabilities           -->
      <!-- ================================================= -->
      <sml:capabilities>
        <swe:DataRecord>
          <gml:name>Pointing Capabilities</gml:name>
          <swe:field name="Across-Track Pointing Range">
            <swe:QuantityRange definition="urn:x-ogc:def:phenomenon:ESA:acrossTrackPointingRange">
              <gml:description>Maximum pointing angles in the across-track direction</gml:description>
              <swe:uom code="deg"/>
              <swe:value>-0 +0</swe:value>
            </swe:QuantityRange>
          </swe:field>
          <swe:field name="Along-Track Pointing Range">
            <swe:QuantityRange definition="urn:x-ogc:def:phenomenon:ESA:alongTrackPointingRange">
              <gml:description>Maximum pointing angles in the along-track direction</gml:description>
              <swe:uom code="deg"/>
              <swe:value>-0 +0</swe:value>
            </swe:QuantityRange>
          </swe:field>
        </swe:DataRecord>
      </sml:capabilities>
      <!-- ================================================= -->
      <!--              Relevant Contacts                    -->
      <!-- ================================================= -->
      <sml:contact xlink:role="urn:x-ogc:def:dictionary:OGC:contactTypes:v01#operator">
        <sml:ResponsibleParty>
          <sml:individualName>Didier Giacobbo</sml:individualName>
          <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>
      <!-- ========================================================== -->
      <!--              System Documentation              -->
      <!-- ========================================================== -->
      <sml:documentation xlink:role="urn:x-ogc:def:dictionary:OGC:documentTypes:v01#datasheet">
        <sml:Document>
          <gml:description>Page of CNES Website describing the HRS Instrument</gml:description>
          <sml:onlineResource xlink:href="http://spot5.cnes.fr/gb/satellite/camerasHRS.htm"/>
        </sml:Document>
      </sml:documentation>
      <!-- ========================================================== -->
      <!--                 System Inputs                -->
      <!-- ========================================================== -->
      <sml:inputs>
        <sml:InputList>
          <sml:input name="Radiation">
            <swe:ObservableProperty definition="urn:x-ogc:def:phenomenon:OGC:radiation"/>
          </sml:input>
        </sml:InputList>
      </sml:inputs>
      <!-- ========================================================== -->
      <!--                 System Outputs               -->
      <!-- ========================================================== -->
      <sml:outputs>
        <sml:OutputList>
          <sml:output name="Foreview Image">
            <swe:DataArray>
              <swe:elementCount>
                <swe:Count>
                  <swe:value>12000</swe:value>
                </swe:Count>
              </swe:elementCount>
              <swe:elementType name="ScanLine">
                <swe:DataArray>
                  <swe:elementCount>
                    <swe:Count>
                      <swe:value>12000</swe:value>
                    </swe:Count>
                  </swe:elementCount>
                  <swe:elementType name="Pixel Value">
                    <swe:Count definition="urn:x-ogc:def:phenomenon:OGC:radiance"/>
                  </swe:elementType>
                </swe:DataArray>
              </swe:elementType>
            </swe:DataArray>
          </sml:output>
          <sml:output name="Rearview Image">
            <swe:DataArray>
              <swe:elementCount>
                <swe:Count>
                  <swe:value>12000</swe:value>
                </swe:Count>
              </swe:elementCount>
```

```
              <swe:elementType name="ScanLine">
                <swe:DataArray>
                  <swe:elementCount>
                    <swe:Count>
                      <swe:value>12000</swe:value>
                    </swe:Count>
                  </swe:elementCount>
                  <swe:elementType name="Pixel Value">
                    <swe:Count definition="urn:x-ogc:def:phenomenon:OGC:radiance"/>
                  </swe:elementType>
                </swe:DataArray>
              </swe:elementType>
            </swe:DataArray>
          </sml:output>
        </sml:OutputList>
      </sml:outputs>
      <!-- -->
      <sml:components>
        <sml:ComponentList>
          <!-- ====================================================== -->
          <!--          HRS1 Detector Description            -->
          <!-- ====================================================== -->
          <sml:component name="HRS1 Detector">
            <sml:Component>
              <!-- -->
              <gml:description>
                In the HRS1, a bar of 12000 CCD detectors is used to acquire the foreview scanlines.
              </gml:description>
              <!-- -->
              <sml:identification>
                <sml:IdentifierList>
                  <sml:identifier name="Short Name">
                    <sml:Term definition="urn:x-ogc:def:identifier:OGC:shortName">
                      <sml:value>HRS1</sml:value>
                    </sml:Term>
                  </sml:identifier>
                  <sml:identifier name="Long Name">
                    <sml:Term definition="urn:x-ogc:def:identifier:OGC:longName">
                      <sml:value>Panchromatic HRS1 Detector</sml:value>
                    </sml:Term>
                  </sml:identifier>
                  <sml:identifier name="Band ID">
                    <sml:Term definition="urn:x-ogc:def:identifier:OGC:bandId">
                      <sml:value>ForeView</sml:value>
                    </sml:Term>
                  </sml:identifier>
                </sml:IdentifierList>
              </sml:identification>
              <!-- -->
              <sml:characteristics>
                <swe:DataRecord>
                  <gml:name>Geometric Characteristics</gml:name>
                  <swe:field name="Across-Track Ground Resolution">
                    <swe:Quantity definition="urn:x-ogc:def:phenomenon:ESA:acrossTrackGroundResolution">
                      <gml:description>Ground sampling resolution in the across-track direction at
nadir</gml:description>
                      <swe:uom code="m"/>
                      <swe:value>10</swe:value>
                    </swe:Quantity>
                  </swe:field>
                  <swe:field name="Along-Track Ground Resolution">
                    <swe:Quantity definition="urn:x-ogc:def:phenomenon:ESA:alongTrackGroundResolution">
                      <gml:description>Ground sampling resolution in the along-track direction at 
nadir</gml:description>
                      <swe:uom code="m"/>
                      <swe:value>5</swe:value>
                    </swe:Quantity>
                  </swe:field>
                  <swe:field name="Number of Samples">
                    <swe:Count definition="urn:x-ogc:def:data:ESA:numberOfSamples">
                      <gml:description>Number of samples collected for this band</gml:description>
                      <swe:value>12000</swe:value>
                    </swe:Count>
                  </swe:field>
                </swe:DataRecord>
              </sml:characteristics>
              <!-- -->
              <sml:characteristics>
                <swe:DataRecord>
                  <gml:name>Measurement Characteristics</gml:name>
                  <swe:field name="Band Type">
                    <swe:Category definition="urn:x-ogc:def:classifier:OGC:bandType">
                      <swe:codeSpace xlink:href="urn:x-ogc:def:dictionary:ESA:bandTypes:v01"/>
                      <swe:value>VIS</swe:value>
                    </swe:Category>
                  </swe:field>
                  <swe:field name="Spectral Range">
                    <swe:QuantityRange definition="urn:x-ogc:def:phenomenon:ESA:spectralRange">
                      <gml:description>Nominal Spectral Range of this detector</gml:description>
                      <swe:uom code="nm"/>
                      <swe:value>490 690</swe:value>
                    </swe:QuantityRange>
                  </swe:field>
                  <swe:field name="SNR Ratio">
                    <swe:Quantity definition="urn:x-ogc:def:phenomenon:ESA:snrRatio">
                      <gml:description>Signal to Noise ratio of this detector</gml:description>
                      <swe:value>120</swe:value>
                    </swe:Quantity>
                  </swe:field>
                </swe:DataRecord>
              </sml:characteristics>
              <!-- -->
            </sml:Component>
          </sml:component>
          <!-- ================================================= -->
          <!--          HRS2 Detector Description           -->
          <!-- ================================================= -->
          <sml:component name="HRS2 Detector">
            <sml:Component>
              <!-- -->
              <gml:description>
                In the HRS2, a bar of 12000 CCD detectors is used to acquire the afterview scanlines.
              </gml:description>
              <!-- -->
              <sml:identification>
                <sml:IdentifierList>
                  <sml:identifier name="Short Name">
                    <sml:Term definition="urn:x-ogc:def:identifier:OGC:shortName">
                      <sml:value>HRS2</sml:value>
                    </sml:Term>
                  </sml:identifier>
                  <sml:identifier name="Long Name">
                    <sml:Term definition="urn:x-ogc:def:identifier:OGC:longName">
                      <sml:value>Panchromatic HRS2 Detector</sml:value>
                    </sml:Term>
                  </sml:identifier>
                  <sml:identifier name="Band ID">
                    <sml:Term definition="urn:x-ogc:def:identifier:OGC:bandId">
                      <sml:value>AfterView</sml:value>
                    </sml:Term>
                  </sml:identifier>
                </sml:IdentifierList>
              </sml:identification>
              <!-- -->
              <sml:characteristics>
                <swe:DataRecord>
                  <gml:name>Geometric Characteristics</gml:name>
                  <swe:field name="Across-Track Ground Resolution">
                    <swe:Quantity definition="urn:x-ogc:def:phenomenon:ESA:acrossTrackGroundResolution">
                      <gml:description>Ground sampling resolution in the across-track direction at
          nadir</gml:description>
                      <swe:uom code="m"/>
                      <swe:value>10</swe:value>
                    </swe:Quantity>
                  </swe:field>
                  <swe:field name="Along-Track Ground Resolution">
                    <swe:Quantity definition="urn:x-ogc:def:phenomenon:ESA:alongTrackGroundResolution">
                      <gml:description>Ground sampling resolution in the along-track direction at
          nadir</gml:description>
                      <swe:uom code="m"/>
                      <swe:value>5</swe:value>
                    </swe:Quantity>
                  </swe:field>
                  <swe:field name="Number of Samples">
                    <swe:Count definition="urn:x-ogc:def:data:ESA:numberOfSamples">
                      <gml:description>Number of samples collected for this band</gml:description>
                      <swe:value>12000</swe:value>
                    </swe:Count>
                  </swe:field>
                </swe:DataRecord>
              </sml:characteristics>
              <!-- -->
              <sml:characteristics>
                <swe:DataRecord>
                  <gml:name>Measurement Characteristics</gml:name>
                  <swe:field name="Band Type">
                    <swe:Category definition="urn:x-ogc:def:classifier:OGC:bandType">
                      <swe:codeSpace xlink:href="urn:x-ogc:def:dictionary:ESA:bandTypes:v01"/>
                      <swe:value>VIS</swe:value>
                    </swe:Category>
                  </swe:field>
                  <swe:field name="Spectral Range">
                    <swe:QuantityRange definition="urn:x-ogc:def:phenomenon:ESA:spectralRange">
                      <gml:description>Nominal Spectral Range of this detector</gml:description>
                      <swe:uom code="nm"/>
                      <swe:value>490 690</swe:value>
                    </swe:QuantityRange>
                  </swe:field>
                  <swe:field name="SNR Ratio">
                    <swe:Quantity definition="urn:x-ogc:def:phenomenon:ESA:snrRatio">
                      <gml:description>Signal to Noise ratio of this detector</gml:description>
                      <swe:value>120</swe:value>
                    </swe:Quantity>
                  </swe:field>
                </swe:DataRecord>
              </sml:characteristics>
```

```
              <!-- -->
            </sml:Component>
          </sml:component>
        </sml:ComponentList>
      </sml:components>
    </sml:System>
  </sml:member>
</sml:SensorML>
```