Open Geospatial Consortium

Date: 2011-03-28

Reference number of this OGC® project document: **OGC 10-135**

OGC name of this OGC® project document: **http://www.opengis.net/doc/IS/EOSPS/2.0**

Version: 2.0

Category: OGC® Interface Standard

Editors: Alexandre Robin (Spot Image)
Philippe Mérigot (Spot Image)

OGC® Sensor Planning Service Interface Standard 2.0
Earth Observation Satellite Tasking Extension

Warning

This document is an OGC Member approved international standard. This document is available on a royalty free, non-discriminatory basis. Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Contents

Page

Figures

i. Abstract

The SPS 2.0 Earth Observation Satellite Tasking Extension Standard specifies extensions to the OGC Sensor Planning Service (SPS) 2.0 Interface Standard. The SPS configuration proposed in this extension is intended to support the programming process of Earth Observation (EO) sensor systems. This standard 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 resulting extended web service interface can be used 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 canceling such a request, and for requesting information on means of obtaining the data collected by the requested task.

ii. Keywords

ogcdoc, sps, earth observation, hma, eo

iii. Preface

This document defines an extension of the version 2.0 of the Sensor Planning Service (SPS) Interface Standard. It is applicable to the tasking of earth observation satellites.

This standard has been developed in the context of the Heterogeneous Mission Accessibility (HMA) project initiated by European Space Agency (ESA). The goal of this extension is to define a coherent web service interface for sending requests for future acquisition of data products to various types of space borne earth observation systems.

Suggested additions, changes, and comments on this draft report are welcome and encouraged. Such suggestions may be submitted by email message or by making suggested changes in an edited copy of this document.

iv. Document terms and definitions

This document uses the standard terms defined in [OGC 06-121], which is based on the ISO/IEC Directives, Part 2. Rules for the structure and drafting of International Standards. In particular, the word “shall” (not “must”) is the verb form used to indicate a requirement to be strictly followed to conform to this standard.

v. Submitting organizations

The following organizations submitted this document to the Open Geospatial Consortium Inc.

- **ESA – European Space Agency**
- **Spot Image S.A.**
- **Deimos Space S.L.U.**
- **Spacebel S.A.**
- **EADS Astrium**

vi. Document contributor contact points

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

Name	Organization	Contribution
Alexandre Robin	Spot Image	Editor
Philippe Mérigot	Spot Image	Editor
Reuben Wright	Deimos Space	ATS, review and comments
Daniele Marchionni	DATAMAT	Review and comments
Jolyon Martin	ESA	Review and comments
Patrick Floissac	Magellium (for CNES)	Review and comments
Ingo Simonis	iGSI	Review and comments

vii. Changes to the OGC Abstract Specification

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

viii. Future work

Additional extensions are planned to support other types of earth observation instruments such as atmospheric and scientific missions. Improvements to this standard are desirable to address interferometric SAR acquisitions.

Foreword

This document is an Earth Observation extension for the OGC Sensor Planning Service (SPS) Interface Standard version 2.0. It defines data models as well as additional operations that can be used to provide tasking capabilities of space borne earth observation systems while still being compatible with the SPS 2.0 standard.

This second edition cancels and replaces the best practice document OGC 07-018r2 titled "Sensor Planning Service Application Profile for EO Sensors 0.9.5". The editorial content of the document has been improved and technically revised.

This document references several external standards and specifications as dependencies (See the normative reference table in section 3 for details). The main dependencies are on the OGC standards listed below:

a) Sensor Planning Service Interface Standard v2.0, [OGC 09-000]

b) SWE Common Data Model Encoding Standard v2.0, [OGC 08-094]

c) Earth Observation Metadata profile of Observations & Measurements, [OGC 10-157157]

d) SWE Service Model Standard v2.0, [OGC 09-001]

e) OGC® Web Services Common Specification v2.0, [OGC 06-121]

This document includes three annexes; Annexes A and B are normative, and annex C is informative.

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

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

Introduction

The SPS configuration proposed in this extension is intended to support the programming process of Earth Observation (EO) sensor systems. This standard 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 EO-SPS is designed to be flexible enough to handle the variety of configurations necessary to cover most EO satellite programming needs.

This document is divided in several requirements class defining requirements for client and server implementations in terms of the structure of the XML exchanged as well as in terms of behavior:

- The first requirements class in section 7.1 constitutes the core of this standard and defines all mandatory components that any implementation of this standard should provide.
- Sections 7.2 and 7.3 define additional requirements that improve support of optical and SAR earth observation systems respectively.
- Section 8 contains requirements classes concerning additional operations that provide functionalities specific to satellite tasking and thus not included in the Sensor Planning Service standard.
- Section 9 contains requirements classes regarding further specification of the Sensor Planning Service state transition and their impact on satellite acquisition status codes, and resulting notifications.

1 Scope

This standard specifies interfaces and parameters that are extensions of the Sensor Planning Service 2.0 Interface Standard [OGC 09-000]. These extensions are dedicated to providing an interoperable access to the tasking capabilities of various types of earth observation systems. The resulting extended web service interface can be used 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 on means of obtaining the data collected by the requested task.

This document builds on information models, descriptions and information defined in version 2.0 of the Sensor Planning Service standard. In particular this extension extends or defines operations for:

- Getting the list of parameters that can be specified for programming a specific space borne earth observation instrument
- Verifying the feasibility of a request that is going to be submitted
- Submitting the request and checking its progress
- Subscribing and receiving notifications about a task's progress
- If necessary canceling or updating the submitted request
- Getting information allowing the ordering or retrieval of the data acquired by the sensor

2 Compliance

Compliance with this standard shall be checked using all the relevant tests specified in Annex A. This annex is normative.

The requirements classes of these standards all have the same standardization target type: server implementation. The conformance tests provided in the abstract test suite in Annex A thus address only this target type. However, most requirements are worded in a way that they also give clear guidance to client implementers even though no conformance tests are formally defined for this target type.

Implementations seeking conformance to this standard shall pass at least the first conformance test class (A.2) and can also decide to pass other optional conformance test classes.

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.

[W3C XML]	W3C Recommendation 6 October 2000, Extensible Markup Language (XML) 1.0, http://www.w3.org/TR/REC-xml
[W3C XNS]	W3C Recommendation January 1999, Namespaces In XML http://www.w3.org/TR/2000/REC-xml-names
[W3C XSD0]	W3C Recommendation 2 May 2001: XML Schema Part 0: Primer http://www.w3.org/TR/2001/REC-xmlschema-0-20010502/
[W3C XSD1]	W3C Recommendation 2 May 2001: XML Schema Part 1: Structures http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/
[W3C XSD2]	W3C Recommendation 2 May 2001: XML Schema Part 2: Datatypes http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/
[W3C SOAP]	W3C Recommendation (24 June 2003): SOAP Version 1.2 Part 1: Messaging Framework, http://www.w3.org/TR/SOAP/
[W3C WSDL]	WSDL, Web Services Description Language (WSDL) 1.1 http://www.w3.org/TR/wsdl
[W3C WS-A]	Web Services Addressing (WS-Addressing) W3C Member Submission 10 August 2004
[OASIS WSN]	Web Service Base Notification 1.3 (WS-Base Notification) OASIS standard, 1 October 2006
[OGC 06-121]	OGC® 06-121r9 OGC® Web Services Common Specification v2.0
[OGC 08-094]	OGC® 08-094r1 SWE Common Data Model Encoding Standard v2.0
[OGC 09-001]	OGC® 09-001 SWE Service Model Standard v2.0
[OGC 10-157]	OGC® 10-157 Earth Observation Metadata profile of Observations & Measurements
[OGC 09-000]	OGC® 09-000r1 Sensor Planning Service Interface Standard v2.0
[OGC 07-036]	OGC® 07-036 Geographic Markup Language (GML) Encoding Standard v3.2.1
[UCUM]	UCUM, Unified Code for Units of Measure http://aurora.rg.iupui.edu/UCUM
[ISO 19107]	ISO 19107:2003 Geographic Information – Spatial Schema

4 Terms and definitions

For the purposes of this standard, the definitions specified in Clause 4 of the OWS Common Implementation Specification [OGC 06-121] and Sensor Planning Service Interface Standard [OGC 09-000] 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. Earth Observation Satellite

Artificial spacecraft specifically designed to observe the earth from orbit, often using sensors that are sensitive to some part of the electromagnetic spectrum.

4.3. 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

4.4. Requirement

Something that is necessary in advance

4.5. State

Condition that persists for a period

5 Conventions

5.1 Abbreviated terms

Most of the abbreviated terms listed in Subclause 5.1 of the OWS Common 2.0 standard [OGC 06-121] apply to this document, plus the following abbreviated terms.

API	Application Program Interface
ATM	Atmospheric
COTS	Commercial Off The Shelf
CRS	Coordinate Reference System
CSW	Catalogue Service-Web
EO	Earth Observation
EO-SPS	Earth Observation Satellite Tasking Extension for the SPS 2.0 Standard
GML	Geographic Markup Language
HMA	Heterogeneous Missions Accessibility
HTTP	Hypertext Transport Protocol
ISO	International Organization for Standardization
OGC	Open Geospatial Consortium
OPT	Optical
QoS	Quality of Service
SAR	Synthetic Aperture Radar
SensorML	Sensor Model Language
SOAP	Simple Object Access Protocol
SPS	Sensor Planning Service (Interface Standard)
SQL	Structured Query Language
SWE	Sensor Web Enablement

UCUM Unified Code for Units of Measure

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

Most diagrams that appear in this standard are presented using the Unified Modeling Language (UML) static structure diagram, as described in Subclause 5.2 of [OGC 06-121].

5.3 Used parts of other documents

This document may use significant parts of the OGC SPS 2.0 standard [OGC 09-000]. To reduce the need to refer to these documents, this document copies some of those parts with small modifications. To indicate those parts to readers of this document, the largely copied parts are shown with a light grey background (15%).

5.4 Platform-neutral and platform-specific standards

As specified in Clause 10 of OGC Abstract Specification Topic 12 “OpenGIS Service Architecture” (which contains ISO 19119), this document includes both Distributed Computing Platform-neutral and platform-specific standards. This document first specifies each operation request and response in platform-neutral fashion. This is done using a table for each data structure, which lists and defines the parameters and other data structures contained. These tables serve as data dictionaries for the UML model presented in Subclauses named “Data Model” throughout the document, and thus specify the UML model data type and multiplicity of each listed item.

EXAMPLES 1 Platform-neutral standards are contained in Subclauses 7.1.2.2.2, 7.1.2.3.2, 7.1.2.4.2, 7.1.2.5.2, 7.1.2.6.2., 7.1.2.7.2, 7.1.2.8.2, etc.

The specified platform-neutral data could be encoded in many alternative ways, each appropriate to one or more specific DCPs. This document now specifies encoding

appropriate for use of HTTP POST transfer of operations requests (using XML encoding). The same XML encoded operation requests and responses can be encoded for other specific computing platforms, including the optional SOAP binding defined in clause 8.4. XML encoding of request and responses are defined in Subclauses named "SWE Common encoding" or "XML encoding" throughout the document.

EXAMPLES 2 Platform-specific standards for XML encoding are contained in Subclauses 7.1.2.3.3, 7.1.2.4.3, 7.1.2.5.3, 7.1.2.6.3, 7.1.2.7.3, 7.1.2.9.3, 7.1.3.1.2, 7.1.3.2.2, 7.1.4.2.3, etc.

5.5 Table notation used to express requirements

Requirements class and individual requirements are clearly highlighted and identified throughout the document by using tables and URL identifiers. Each requirements class is marked with the tabular format shown below:

Requirements Class	
http://www.opengis.net/spec/EOSPS/2.0/req/{class-name}	
Target Type	Description of standardization target type
Dependency	**http://www.opengis.net/spec/EOSPS/2.0/req/{class-name}**

Individual requirements are described in the following tabular format:

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/{class-name}/{req-name}
Req N. Textual description of requirement.

6 EO SPS overview

The SPS operations can be divided into informational and functional operations. The informational operations are the *GetCapabilities*, *DescribeSensor*, *DescribeTasking*, *DescribeResultAccess* and *GetStatus* operation. The functional operations are the *GetFeasibility*, *Reserve*, *Commit*, *Submit*, *Update* and *Cancel* operations. Another informational operation, *GetSensorAvailability*, and two other functional operations *Validate* and *SubmitSegmentByID* are defined in this EO extension.

Existing SPS operations that are inherited by this standard are described below (gray background):

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 sensor identifiers provided by the service.

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

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

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

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

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

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

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

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 an OGC Web services nevertheless OGC Web services such as SOS, WMS, WFS or WCS are desirable.

The following operations enable clients to reserve a task instead of directly submitting it. This facilitates tasking of a group of SPSs. Reserved tasks have a finite lifetime before they expire. During this lifetime such a task can be confirmed so that the service starts execution.

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

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

This standard defines additional content that is to be used within some of the existing SPS operations listed above:

- Extensions to the SPS *StatusReport* object are defined to provide additional information regarding feasibility results and monitoring acquisition progress. These extended objects are defined in sections 7.1.4 and 7.1.5 and are used within responses of the *GetFeasibility*, *Submit* and *GetStatus* operations.
- Simple extensions to tasking requests are defined in sections 7.1.3. They are used within *GetFeasibility*, *Submit* and *Reserve* requests.

This standard also defines the additional operations listed below:

GetSensorAvailability (optional) – This operation provides information on the availability of the sensor during certain time periods.

Validate (optional) – Several acquisition attempts are sometimes necessary to obtain a satisfying result (case of optical acquisitions over cloudy areas for example). The Validate operation can be used by the client to indicate that an acquisition is satisfactory and thus to stop collecting new images for this area.

SubmitSegmentByID (optional) – This operation allows a client to submit a task for acquisition of only certain segments listed in a feasibility study, by using their ID. *Note that this operation would usually only be implemented by systematic missions.*

These operations have many similarities to other OGC® Web Services. Many of these interface aspects that are common with other OWSs are thus specified in the OpenGIS® Web Services Common Implementation Specification [OGC 06-121]. Many of these common aspects are normatively referenced herein, instead of being repeated in this specification.

7 Extensions to existing SPS operations

7.1 Requirements Class: Core requirements for all mission types

Requirements Class	
http://www.opengis.net/spec/EOSPS/2.0/req/core	
Target Type	Server Implementation
Dependency	http://www.opengis.net/doc/IS/SPS/2.0/clause/7
Dependency	http://www.opengis.net/doc/IS/SPS/2.0/clause/8
Dependency	http://www.opengis.net/doc/IS/SPS/2.0/clause/10

7.1.1 Introduction

This clause details core requirements that all implementations of the Sensor Planning Service supporting the "Earth Observation Satellite Tasking Extension" (i.e. EO SPS) have to fulfill. Implementations of EO SPS servers for all mission types shall satisfy the requirements of this class. Most requirements are also applicable to client implementations even though no formal conformance testing is defined for clients.

Since this standard is an extension of the Sensor Planning Service 2.0 standard, these implementations shall also satisfy requirements of the SPS 2.0 standard.

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/core/dependency-sps
Req 1. An EO SPS implementation shall first pass the "Core", "Feasibility Controller" and "XML Encoding" conformance test classes of the "Sensor Planning Service Interface Standard 2.0" standard [OGC 09-000].

7.1.2 Tasking parameters

This paragraph defines the list of parameters a client has to provide to task an EO satellite. The objective is to define the name, type and units of the parameters shared by all the EO missions in a multi mission context. This list can be easily extended with parameters that are specific to certain missions or data providers.

As required by the SPS 2.0 standard, these parameters are described in the *DescribeTaskingResponse* document generated by the server, thus providing a schema that the client has to use to correctly formulate tasking requests. Only tasking parameter values are then sent in tasking requests (see [OGC 09-000] for more information).

For interoperability reason, when implementing an EO SPS, developers shall use (some of) the tasking parameters specified in this document, even if the name or the unit is different than the ones they usually use. Implementers should thus expect conversions when mapping these standardized field values to the input of existing ground segment planning systems.

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/core/tasking-params-valid
Req 2. The *DescribeTaskingResponse* document shall contain the tasking parameters specified in this document encoded in SWE Common by using exclusively the field names, definition and units specified in the tables of this specification.

All or part of these EO parameters can be used by a particular EO SPS instance. Each of them can be made optional or mandatory and its value can be further restricted (by interval or enumerated values).

A given EO SPS instance can also insert vendor, mission or sensor specific parameters in the list of tasking parameters advertised by the server, in addition to those defined in this standard. Such parameters shall be used only if no equivalent parameter is defined in this standard and shall always be optional in order to be compatible with clients that have no prior knowledge of the mission details.

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/core/tasking-params-vendor
Req 3. Additional vendor parameters shall be used only when no equivalent parameter is defined in this standard and shall be marked as optional.

Note: it is recommended that all parameters marked as optional have a default value in order to inform the client of what value will be used if the parameter is omitted.

7.1.2.1 SWE Common encoding

As specified by the SPS 2.0 standard, the description of the tasking parameters shall be returned in the *DescribeTasking* operation response and implemented using the SWE Common format defined in [OGC 08-094]. The *DescribeTaskingResponse* XML document thus contains a list of tasking parameters definitions that are a subset of the parameters defined in this section.

7.1.2.1.1 Default parameter values

The description of a tasking parameter may contain a default value, specified in a *swe:value* element:

```
<swe:field name="FusionAccepted">
  <swe:Boolean definition="http://www.opengis.net/def/property/OGC-EO/0/FusionAccepted"
    optional="true">
    <swe:value>true</swe:value>
  </swe:Boolean>
</swe:field>
```

In the example above, the default value for the *FusionAccepted* parameter is *true*. It means that if the client does not specify a value, the server will automatically apply the value *true* for this parameter.

7.1.2.1.2 Values of parameters embedded in tasking requests

Values of tasking parameters are encoded with any of the encoding defined in the SWE Common standard when embedded within a tasking request (i.e. *Submit*, *Update*, *GetFeasibility*, *Reserve*, etc.). A compliant EO SPS service implementation shall support at least the XML encoding method (identified by the corresponding URI defined in the SPS 2.0 standard).

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/core/tasking-params-xml-encoding
Req 4. An EO SPS implementation shall support the *XMLEncoding* defined in clause 8.5 of [OGC 08-094] "SWE Common Data Model 2.0 Encoding Standard" and pass the corresponding conformance tests.

Please refer to [OGC 08-094] for details of the XML encoding rules. Annex C contains examples of tasking requests with the matching tasking parameters description.

As required by the SPS interface standard [OGC 09-000], any invalid value for a tasking parameter (i.e. not matching the parameter definitions provided in *DescribeTaskingResponse*) shall result in an *InvalidParameterException* being sent back to the client.

7.1.2.2 ProgrammingRequest Class

7.1.2.2.1 Description

The *ProgrammingRequest* class is the root of the list of tasking parameters. It is abstract and cannot be instantiated. Instead, the *CoverageProgrammingRequest* class (cf. §7.1.2.4) or the *SwathProgrammingRequest* class (cf. §7.1.2.5) are used.

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/core/tasking-params-root
Req 5. The root of the list of tasking parameters (in *DescribeTaskingResponse*) shall be a concrete instance of the *ProgrammingRequest* class.

7.1.2.2.2 Data model

This abstract class contains two common parameter groups that are inherited by its descendants. These groups are represented by the *QualityOfService* and *ValidationParameters* classes, as shown in the UML diagram below:

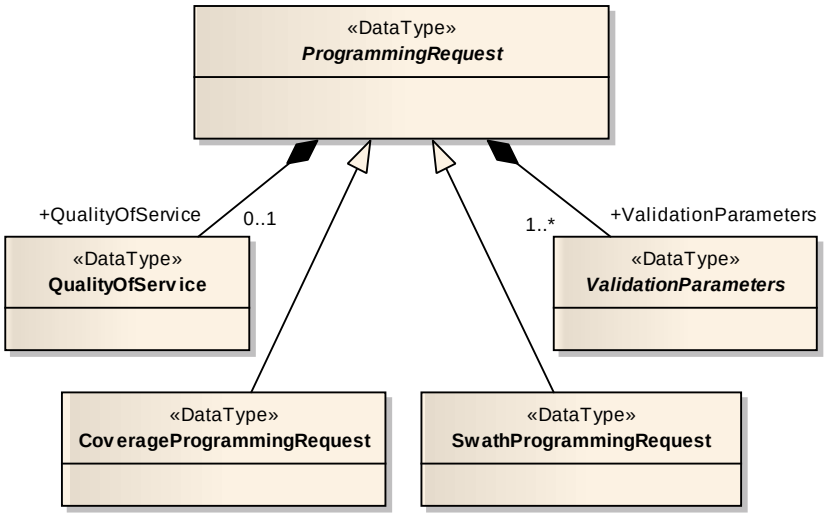

Figure 1 – UML diagram of the *ProgrammingRequest* class

Classes represented in this diagram are fully described in later sections:

The class *QualityOfService* is specified in clause §7.1.2.3.

The class *CoverageProgrammingRequest* is specified in clause §7.1.2.4.

The class *SwathProgrammingRequest* is specified in clause §7.1.2.5.

The class *ValidationParameters* is specified in clause §7.1.2.16.

7.1.2.3 QualityOfService Class

7.1.2.3.1 Description

This parameter group is used to specify the level of priority of the programming request. In general, the user should expect more reactivity from a 'HIGH' priority request than for a 'STANDARD' priority request. However, he will usually be charged more in order to obtain this high priority service. The exact meaning of the priority levels have to be agreed upon with the provider and are usually defined verbally in its terms of service. If the EO-SPS instance implements an authentication method, the internal priority level can be defined on a per user basis by associating it to the user account.

If this tasking parameter is supported, the EO-SPS service shall allow at least the two values 'STANDARD' and 'HIGH' but may provide other mission specific priority levels as well.

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/core/priority-default
Req 6. The *Priority* field shall be of type enumeration with at least the two default priority levels 'STANDARD' and 'HIGH'.

Note: A request with a high level of priority can be rejected by the server if the user authorizations don't allow it (i.e. a user with no special permissions may not be able to get high priority even though he requested it).

All vendor/mission specific extensions related to quality of service should be inserted into this group.

7.1.2.3.2 Data model

This parameter group contains a single *PriorityLevel* attribute but it provides a container for additional mission specific parameters:

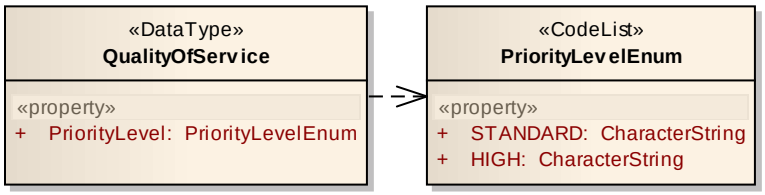

Figure 2 – UML diagram of the *QualityOfService* class

Name	Description	Type (unit)
PriorityLevel	Requested priority level: STANDARD or HIGH, + other mission specific levels (e.g. P0, P1 in the example below)…	CodeList STANDARD, HIGH

7.1.2.3.3 SWE Common encoding

The following table and XML snippet describe how this tasking parameter is to be encoded using the SWE Common Data Model. Even though this standard defines a single parameter related to quality of service (i.e. Priority), the *QualityService* class is encoded as a *DataRecord* to allow the insertion of future quality related parameters:

Field Name	Definition	Component
QualityOfService	http://www.opengis.net/def/property/OGC-EO/0/QualityOfService	DataRecord
PriorityLevel	http://www.opengis.net/def/property/OGC-EO/0/PriorityLevel	Category

In the following XML snippet, two additional mission specific priority levels have been defined and can thus be used in tasking requests:

```
<swe:field name="QualityOfService">
  <swe:DataRecord definition="http://www.opengis.net/def/property/OGC-EO/0/QualityOfService">
    <gml:name>Quality of Service</gml:name>
    <swe:field name="PriorityLevel">
      <swe:Category definition="http://www.opengis.net/def/property/OGC-EO/0/PriorityLevel">
        <swe:constraint>
          <swe:AllowedTokens>
            <swe:value>STANDARD</swe:value>
            <swe:value>HIGH</swe:value>
            <swe:value>P0</swe:value>
            <swe:value>P1</swe:value>
          </swe:AllowedTokens>
        </swe:constraint>
        <swe:value>STANDARD</swe:value>
      </swe:Category>
    </swe:field>
  </swe:DataRecord>
</swe:field>
```

7.1.2.4 CoverageProgrammingRequest Class

7.1.2.4.1 Description

A *CoverageProgrammingRequest* is used for tasking a satellite based on an area and period of interest. It is thus adapted to end users who may not know the details of the satellite and remote sensing instrument being tasked. It is also useful to allow tasking a constellation of satellites since the *SwathProgrammingRequest* is not applicable in this case. Alternatively, a *SwathProgrammingRequest* can be used (if supported by the server) to request an acquisition from a single satellite based on orbit related criteria (lower level tasking).

7.1.2.4.2 Data model

The following UML diagram shows that in addition to the *QualityOfService* section (inherited from *ProgrammingRequest*), a *CoverageProgrammingRequest* is composed of a *RegionOfInterest*, a *TimeOfInterest* and a list of acquisition parameters (monoscopic or stereoscopic) that are used to express the high level need of the data consumer.

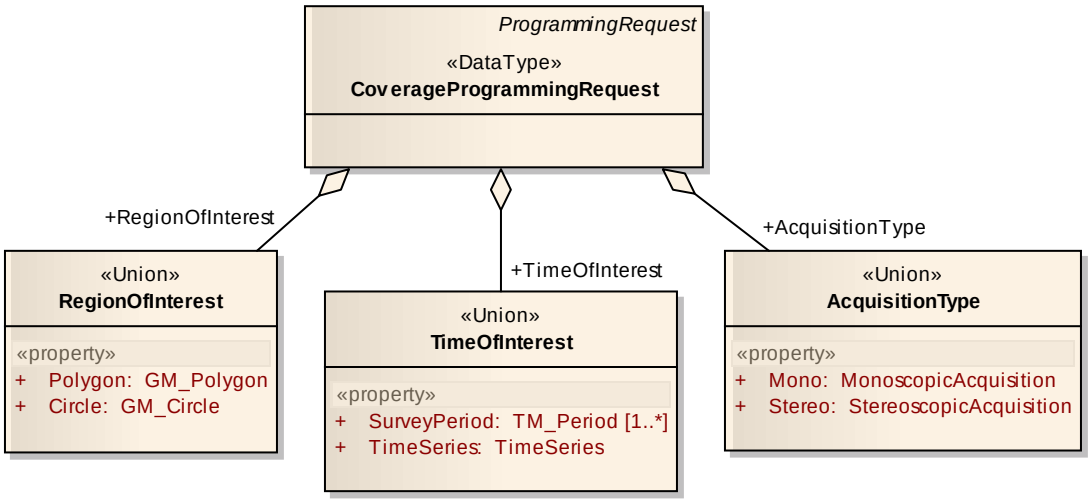

Figure 3 – UML diagram of the *CoverageProgrammingRequest* class

The *RegionOfInterest* class is specified in §7.1.2.6. Only one region of interest can be specified. Multiple requests should be issued if the coverage of several regions is desired.

The *TimeOfInterest* class is specified in §7.1.2.7. Only one time of interest can be specified but can consist of a list of time period if supported by the server.

The *AcquisitionType* class is specified in §7.1.2.9.

Note: Time series are used for requesting multiple acquisitions at different dates, whereas specifying several survey periods means only one acquisition, made during one of the several possible periods.

7.1.2.4.3 SWE Common encoding

The following XML snippet shows the skeleton of tasking parameters to be used for the definition of a *CoverageProgrammingRequest*. The content of each "field" element has been hidden for clarity and is shown in details in later sections:

Field Name	Definition	Component
CoverageProgrammingRequest	http://www.opengis.net/def/property/OGC-EO/0/CoverageProgrammingRequest	DataRecord

```
<DescribeTaskingResponse
  xmlns="http://www.opengis.net/sps/2.0"
  xmlns:swe="http://www.opengis.net/swe/2.0">
  <taskingParameters name="CoverageProgrammingRequest">
    <swe:DataRecord definition="http://www.opengis.net/def/property/OGC-EO/0/CoverageProgrammingRequest">
      <swe:field name="QualityOfService" ... />
      <swe:field name="RegionOfInterest" ... />
      <swe:field name="TimeOfInterest" ... />
      <swe:field name="AcquisitionType" ... />
```



```
    <swe:field name="ValidationParameters" ... />
   </swe:DataRecord>
  </taskingParameters>
</DescribeTaskingResponse>
```

7.1.2.5 SwathProgrammingRequest Class

7.1.2.5.1 Description

A *SwathProgrammingRequest* is used to provide low level tasking capabilities based directly on orbit, track and frame numbers. This is usually restricted to advanced users since it requires a-priori knowledge (i.e. technical knowledge of the orbit characteristics) of the mission being tasked.

Moreover, this tasking mode is not applicable to a "virtual" SPS offering representing several missions and allowing the collaborative tasking of different EO instruments on several platforms. This is because a coherent multi-mission programming request is not possible when supplying orbit related information, each mission having a different orbit cycle. A *SwathProgrammingRequest* is thus inherently single-mission and single-sensor.

7.1.2.5.2 Data model

A *SwathProgrammingRequest* consists of an optional time of interest (i.e. either a simple time period or a time series) and a list of swath segments to acquire. These segments are identified by their orbit, track and frame numbers and are requested to take place either within a precise orbit cycle or at any time during a certain time period. This is shown on the UML diagram below:

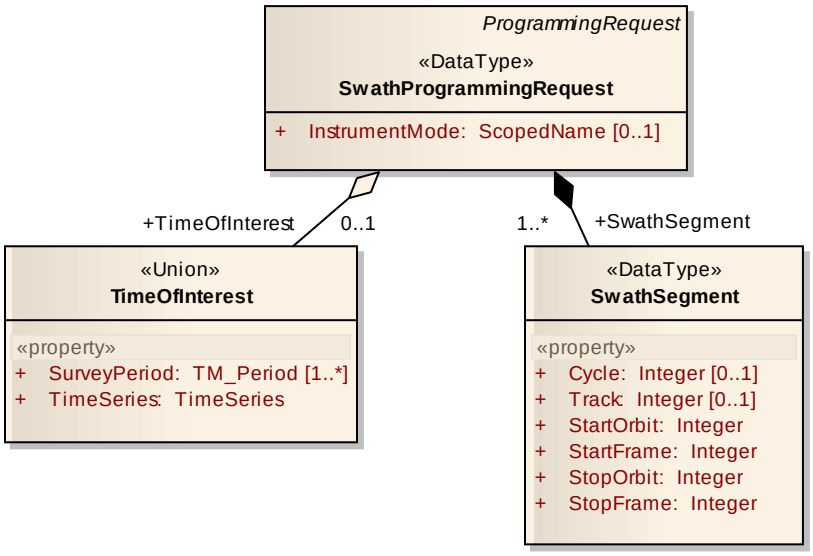

Figure 4 – UML diagram of the *SwathProgrammingRequest* class

When a *TimeOfInterest* is specified, segments can be acquired during any cycle within the temporal window requested. Consequently the *Cycle* attribute of each requested segment shall be ommited. When no *TimeOfInterest* is provided, the *Cycle* attribute shall be included in each segment and the planning system shall try to acquire (or compute the feasibility for) all exact segments requested.

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/core/swath-request-time
Req 7. An instance of the *SwathProgrammingRequest* shall either include a *TimeOfInterest* as the absolute time period of interest when provided, or set the *Cycle* attribute on each individual *SwathSegment* otherwise.

The *TimeOfInterest* class is specified in §7.1.2.7.

A list of swath segments is used to request the acquisition of one or more portions of imagery swath. The table below further describes individual attributes of the *SwathSegment* class:

Name	Description	Type (unit)
Cycle	Absolute orbit cycle number during which to acquire the requested imagery segment. The first cycle usually starts shortly after the platform is launched and it has reached its nominal orbit.	Integer
Track	Number of the track where the acquisition is requested to be done. (This is only necessary for satellites that can be pointed off-nadir or with multiple parallel acquisition swaths).	Integer
StartOrbit	Number of the orbit (within the cycle) where the acquisition is requested to start.	Integer
StartFrame	Number of the frame (within the orbit) where the acquisition is requested to start.	Integer
StopOrbit	Number of the orbit (within the cycle) where the acquisition is requested to stop.	Integer
StopFrame	Number of the frame (within the orbit) where the acquisition is requested to stop.	Integer

7.1.2.5.3 SWE Common encoding

The following table describes how this tasking parameters group shall be encoded using the SWE Common Data Model:

Field Name	Definition	Component
SwathProgramming Request	http://www.opengis.net/def/property/OGC-EO/0/SwathProgrammingRequest	DataRecord
InstrumentMode	http://www.opengis.net/def/property/OGC-EO/0/InstrumentMode	Category
SwathSegmentList	http://www.opengis.net/def/property/OGC-EO/0/SwathSegmentList	DataArray
SwathSegment	http://www.opengis.net/def/property/OGC-EO/0/SwathSegment	DataRecord
Cycle	http://www.opengis.net/def/property/OGC-EO/0/Cycle	Count

Track	http://www.opengis.net/def/property/OGC-EO/0/Track	Count
StartOrbit	http://www.opengis.net/def/property/OGC-EO/0/StartOrbit	Count
StartFrame	http://www.opengis.net/def/property/OGC-EO/0/StartFrame	Count
StopOrbit	http://www.opengis.net/def/property/OGC-EO/0/StopOrbit	Count
StopFrame	http://www.opengis.net/def/property/OGC-EO/0/StopFrame	Count

Values of orbit, track and frame numbers can be further restricted in the *DescribeTaskingResponse* corresponding to a given service offering (i.e. for a given mission). This is shown on the following XML snippet that indicates how a *SwathProgrammingRequest* shall be encoded with the SWE Common Data Model:

```
<DescribeTaskingResponse>
  <taskingParameters name="SwathProgrammingRequest">
    <swe:DataRecord definition="http://www.opengis.net/def/property/OGC-EO/0/SwathProgrammingRequest">
      <swe:field name="InstrumentMode">
        <swe:Category definition="http://www.opengis.net/def/property/OGC-EO/0/InstrumentMode">
          <gml:name>Instrument Mode</gml:name>
          <swe:codeSpace xlink:href="http://www.opengis.net/def/property/OGC-EO/0/opt/SpectralModes/"/>
          <swe:constraint>
            <swe:AllowedTokens>
              <swe:value>PANCHROMATIC</swe:value>
              <swe:value>MULTISPECTRAL</swe:value>
            </swe:AllowedTokens>
          </swe:constraint>
          <swe:value>MULTISPECTRAL</swe:value>
        </swe:Category>
      </swe:field>
      <swe:field name="SwathSegmentList">
        <swe:DataArray definition="http://www.opengis.net/def/property/OGC-EO/0/SwathSegmentList">
          <swe:elementCount>
            <swe:Count/>
          </swe:elementCount>
          <swe:elementType name="SwathSegment">
            <swe:DataRecord definition="http://www.opengis.net/def/property/OGC-EO/0/SwathSegment">
              <swe:field name="Cycle">
                <swe:Count definition="http://www.opengis.net/def/property/OGC-EO/0/Cycle">
                  <gml:name>Cycle Number</gml:name>
                  <swe:constraint>
                    <swe:AllowedValues>
                      <swe:interval>0 +Inf</swe:interval>
                    </swe:AllowedValues>
                  </swe:constraint>
                </swe:Count>
              </swe:field>
              <swe:field name="Track">
                <swe:Count definition="http://www.opengis.net/def/property/OGC-EO/0/Track">
                  <gml:name>Track Number</gml:name>
                  <swe:constraint>
                    <swe:AllowedValues>
                      <swe:interval>1 8</swe:interval>
                    </swe:AllowedValues>
                  </swe:constraint>
                </swe:Count>
              </swe:field>
              <swe:field name="StartOrbit">
                <swe:Count definition="http://www.opengis.net/def/property/OGC-EO/0/StartOrbit">
                  <gml:name>Start Orbit Number</gml:name>
```

```
            <swe:constraint>
              <swe:AllowedValues>
                <swe:interval>1 369</swe:interval>
              </swe:AllowedValues>
            </swe:constraint>
          </swe:Count>
        </swe:field>
        <swe:field name="StartFrame">
          <swe:Count definition="http://www.opengis.net/def/property/OGC-EO/0/StartFrame">
            <gml:name>Stop Frame Number</gml:name>
            <swe:constraint>
              <swe:AllowedValues>
                <swe:interval>1 700</swe:interval>
              </swe:AllowedValues>
            </swe:constraint>
          </swe:Count>
        </swe:field>
        <swe:field name="StopOrbit">
          <swe:Count definition="http://www.opengis.net/def/property/OGC-EO/0/StopOrbit">
            <gml:name>Stop Orbit Number</gml:name>
            <swe:constraint>
              <swe:AllowedValues>
                <swe:interval>1 369</swe:interval>
              </swe:AllowedValues>
            </swe:constraint>
          </swe:Count>
        </swe:field>
        <swe:field name="StopFrame">
          <swe:Count definition="http://www.opengis.net/def/property/OGC-EO/0/StopFrame">
            <gml:name>Stop Frame Number</gml:name>
            <swe:constraint>
              <swe:AllowedValues>
                <swe:interval>1 700</swe:interval>
              </swe:AllowedValues>
            </swe:constraint>
          </swe:Count>
        </swe:field>
      </swe:DataRecord>
    </swe:elementType>
  </swe:DataArray>
</swe:field>
</swe:DataRecord>
</taskingParameters>
</DescribeTaskingResponse>
```

When a *TimeOfInterest* is supported by the service, an additional field is included as shown in the second example below (The content of each field has been omitted for clarity):

```
<DescribeTaskingResponse>
  <taskingParameters name="SwathProgrammingRequest">
    <swe:DataRecord definition="http://www.opengis.net/def/property/OGC-EO/0/SwathProgrammingRequest">
      <swe:field name="TimeOfInterest" ... />
      <swe:field name="InstrumentMode" ... />
      <swe:field name="SwathSegmentList" ... />
    </swe:DataRecord>
  </taskingParameters>
</DescribeTaskingResponse>
```

The *TimeOfInterest* field shall be encoded as specified in clause §7.1.2.7.3.

7.1.2.6 RegionOfInterest Class

7.1.2.6.1 Description

This class is used within a *CoverageProgrammingRequest* to specify the desired geographical region of interest that should be covered by the satellite. A user may specify the *RegionOfInterest* by using either a polygon or a circle. This parameter is mandatory in all EO-SPS tasking requests and shall be supported by all service instances.

7.1.2.6.2 Data model

The *RegionOfInterest* class is a union of the *GM_Polygon* and *GM_Circle* classes defined in [ISO 19107]:

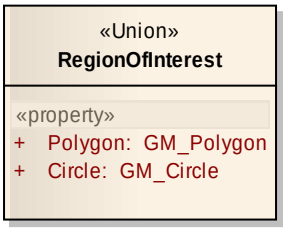

Figure 5 – UML diagram of the *RegionOfInterest* class

7.1.2.6.3 SWE Common encoding of RegionOfInterest

The following table and XML snippet describe how this tasking parameter shall be encoded using the SWE Common Data Model:

Field Name	Definition	Component
RegionOfInterest	http://www.opengis.net/def/property/OGC-EO/0/RegionOfInterest	DataChoice

The following example shows the skeleton of the *RegionOfInterest* parameter definition if the server wants to offer the choice of specifying the ROI via a polygon or a circle:

```
<swe:field name="RegionOfInterest">
  <swe:DataChoice definition="http://www.opengis.net/def/property/OGC-EO/0/RegionOfInterest">
    <gml:name>Region of Interest</gml:name>
    <swe:item name="Polygon" ... />
    <swe:item name="Circle" ... />
  </swe:DataChoice>
</swe:field>
```

Alternatively, it may be too complex or not desired for an EO-SPS to support a choice between polygon and circle. It is then possible for servers to allow the specification of the region of interest by polygon only or circle only. This is done by using a *DataRecord* with a single field instead of a *DataChoice* as shown in the following SWE Common snippet:

```
<swe:field name="RegionOfInterest">
  <swe:DataRecord definition="http://www.opengis.net/def/property/OGC-EO/0/RegionOfInterest">
    <gml:name>Region of Interest</gml:name>
    <swe:field name="Polygon" ... />
  </swe:DataRecord>
</swe:field>
```

The contents of the *Polygon* and *Circle* elements have been hidden for clarity and are detailed in sections §7.1.2.6.4 and §7.1.2.6.5 respectively.

7.1.2.6.4 SWE Common encoding of GM_Polygon

The *GM_Polygon* class shall be encoded in SWE Common as indicated by the table and XML snippet below:

Field Name	Definition	Component
Polygon	http://www.opengis.net/def/objectType/ISO-19107/2003/GM_Polygon	DataRecord
Exterior	http://www.opengis.net/def/objectType/ISO-19107/2003/GM_Ring	DataArray
Point	No definition required but the coordinates unit shall be 'deg' and the reference frame 'http://www.opengis.net/def/crs/EPSG/0/4326' shall be specified	Vector

```
<swe:item name="Polygon">
  <swe:DataRecord definition="http://www.opengis.net/def/objectType/ISO-19107/2003/GM_Polygon">
    <swe:field name="Exterior">
      <swe:DataArray definition="http://www.opengis.net/def/objectType/ISO-19107/2003/GM_Ring">
        <swe:elementCount>
          <swe:Count/>
        </swe:elementCount>
        <swe:elementType name="Point">
          <swe:Vector referenceFrame="http://www.opengis.net/def/crs/EPSG/0/4326">
            <swe:coordinate name="Lat">
              <swe:Quantity definition="http://www.opengis.net/def/property/OGC/0/GeodeticLatitude"
                            axisID="Lat">
                <swe:uom code="deg"/>
              </swe:Quantity>
            </swe:coordinate>
            <swe:coordinate name="Lon">
              <swe:Quantity definition="http://www.opengis.net/def/property/OGC/0/Longitude"
                            axisID="Long">
                <swe:uom code="deg"/>
              </swe:Quantity>
            </swe:coordinate>
          </swe:Vector>
        </swe:elementType>
      </swe:DataArray>
    </swe:field>
  </swe:DataRecord>
</swe:item>
```


7.1.2.6.5 SWE Common encoding of *GM_Circle*

The *GM_Circle* class shall be encoded in SWE Common as indicated by the table and XML snippet below:

Field Name	Definition	Component
Circle	http://www.opengis.net/def/objectType/ISO-19107/2003/GM_Circle	DataRecord
Center	No definition required but the coordinates unit shall be 'deg' and the reference frame 'http://www.opengis.net/def/crs/EPSG/0/4326' shall be specified	Vector
Radius	http://www.opengis.net/def/property/OGC-EO/0/Radius Unit of measure shall be 'km' for kilometers	Quantity

```
<swe:item name="Circle">
  <swe:DataRecord definition="http://www.opengis.net/def/objectType/ISO-19107/2003/GM_Circle">
    <swe:field name="Center">
      <swe:Vector referenceFrame="http://www.opengis.net/def/crs/EPSG/0/4326">
        <swe:coordinate name="Lat">
          <swe:Quantity definition="http://www.opengis.net/def/property/OGC/0/GeodeticLatitude"
                        axisID="Lat">
            <swe:uom code="deg"/>
          </swe:Quantity>
        </swe:coordinate>
        <swe:coordinate name="Lon">
          <swe:Quantity definition="http://www.opengis.net/def/property/OGC/0/Longitude"
                        axisID="Long">
            <swe:uom code="deg"/>
          </swe:Quantity>
        </swe:coordinate>
      </swe:Vector>
    </swe:field>
    <swe:field name="Radius">
      <swe:Quantity definition="http://www.opengis.net/def/property/OGC-EO/0/Radius">
        <swe:uom code="km"/>
      </swe:Quantity>
    </swe:field>
  </swe:DataRecord>
</swe:item>
```

7.1.2.7 TimeOfInterest Class

7.1.2.7.1 Description

This parameter is used to specify the desired period(s) of acquisition. A *TimeOfInterest* may be specified either by one or more time periods *(TM_Period)* or by a single *TemporalSeries* object. At least one of these shall be included in all EO SPS tasking requests and requests with a single survey period shall be supported by all EO-SPS instances:

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/core/survey-period-mandatory
Req 8. The *TimeOfInterest* union shall not be restricted to *TimeSeries* only. A single survey period shall always be a possible option.

Specifying multiple time periods has a different meaning that a time series. In addition to the fact that these periods can be irregularly spaced, a request with several time periods indicates that only one acquisition has to be done within any of the specified time slots. A time series on the other hand corresponds to a request of several acquisitions of the same area at regular intervals. Consequently, only one time series can be specified at a time.

7.1.2.7.2 Data model

The *TimeOfInterest* class conceptually consists of a union of *TM_Period* and *TimeSeries* classes as shown on the following UML diagram:

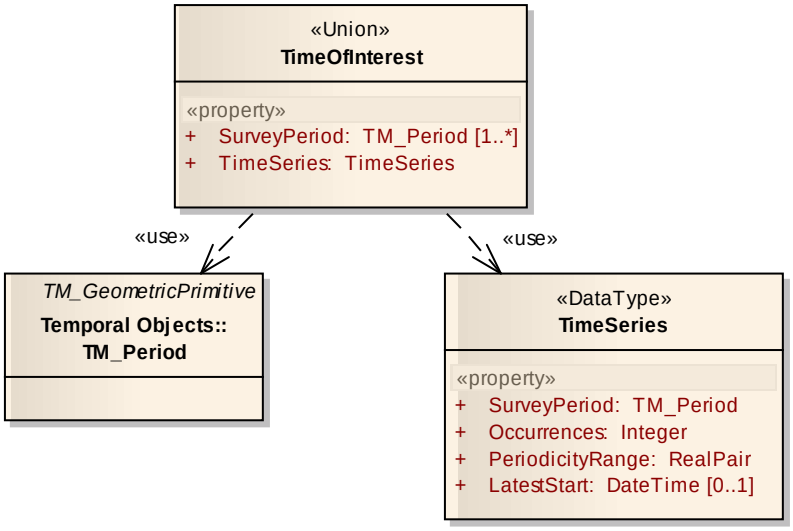

Figure 6 – UML diagram of the *TimeOfInterest* class

The encoding of the *TM_Period* class is specified in clause §7.1.2.7.4.

The class *TimeSeries* is specified in clause §7.1.2.8.

7.1.2.7.3 SWE Common encoding of *TimeOfInterest*

The following table and XML snippet describe how this tasking parameter shall be encoded using the SWE Common Data Model:

Field Name	Definition	Component
TimeOfInterest	http://www.opengis.net/def/property/OGC-EO/0/TimeOfInterest	DataChoice

Below is the skeleton of the *TimeOfInterest* parameter definition:

```
<swe:field name="TimeOfInterest">
  <swe:DataChoice definition="http://www.opengis.net/def/property/OGC-EO/0/TimeOfInterest">
    <gml:name>Time of Interest</gml:name>
    <swe:item name="SurveyPeriod" ... />
    <swe:item name="TimeSeries" ... />
  </swe:DataChoice>
</swe:field>
```

When the support of time series is not desired, a record with a single field containing a survey period shall be used as shown below:

```
<swe:field name="TimeOfInterest">
  <swe:DataRecord definition="http://www.opengis.net/def/property/OGC-EO/0/TimeOfInterest">
    <gml:name>Time of Interest</gml:name>
    <swe:field name="SurveyPeriod" ... />
  </swe:DataRecord>
</swe:field>
```

The contents of the *SurveyPeriod* and *TimeSeries* elements have been hidden for clarity and are detailed in sections §8.1.1.7.4 and §8.1.1.7.5 respectively.

7.1.2.7.4 SWE Common encoding of *TM_Period*

The following table and XML snippet describe how this tasking parameter shall be encoded using the SWE Common Data Model:

Field Name	Definition	Component
SurveyPeriod	http://www.opengis.net/def/property/OGC-EO/0/SurveyPeriod The unit of measure SHALL be 'http://www.opengis.net/def/uom/ISO-8601/0/Gregorian' to indicate that the time will have to be encoded using ISO 8601 units and format.	TimeRange

```
<swe:item name="SurveyPeriod">
  <swe:TimeRange definition="http://www.opengis.net/def/property/OGC-EO/0/SurveyPeriod">
    <gml:name>Survey Period</gml:name>
    <swe:uom xlink:href="http://www.opengis.net/def/uom/ISO-8601/0/Gregorian"/>
  </swe:TimeRange>
</swe:item>
```

7.1.2.8 TimeSeries Class

7.1.2.8.1 Description

This class can be used to specify a time series of acquisitions, that is to say, to request multiple acquisitions of the same area at regular intervals of time.

7.1.2.8.2 Data model

The *TimeSeries* class contains an overall survey period during which all acquisitions must take place and additional attributes that define the number of occurrences, the periodicity and the latest start date of the series. These attributes are described in more details in the table below:

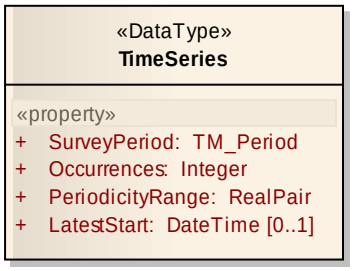

Figure 7 – UML diagram of the *TimeSeries* class

Name	Description	Type (unit)
SurveyPeriod	Overall period during which the whole series should be acquired, while the latest start, if supported, specifies the date before which the first acquisition of the series must be completed.	TM_Period
Occurences	Number of times the region should be fully covered.	Integer
PeriodicityRange*	Acceptable periodicity range for the temporal series (see note below). This allows for a fine grained specification of the time series periodicity ranging from a loose time interval (e.g. between 4 and 8 weeks) to a very precise period (e.g. exactly 10 days).	RealPair (days)
LatestStart	Date before which the first set of acquisitions (i.e. the whole ROI coverage in the case of a *CoverageProgrammingRequest* or all the requested swath segments in the case of a *SwathProgrammingRequest*) should be completed.	DateTime

* The periodicity range provides a flexible mechanism to support strict time series (i.e. with an exact periodicity), as well as series with “frozen days” specifying a minimum duration between two successive acquisition of the series. The period is inclusive.

- An exact periodicity is supported by indicating a tight range. It is specified in number of days. For example the following range [1 1] indicates that the periodicity should be EXACTLY 1 day.

- A number of frozen days can be indicated by only specifying the lower bound of the range. For example, [30 +Inf] means that the period between two successive acquisitions must be AT LEAST 30 days, but can be arbitrary larger, as long as the whole series can be completed within the survey period.

- Finally it is possible to obtain an intermediate behavior by indicating both low and high bounds such as [10 20] meaning that the period between two successive

acquisitions must be BETWEEN 10 and 20 days, thus leaving more flexibility to the satellite operator, when a strict time series is not really required.

The attributes of the *TimeSeries* class shall satisfy an additional constraint since the total duration of the survey period cannot be less than the sum of all intervals between successive acquisitions.

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/core/time-series-coherent
Req 9. The values of the *SurveyPeriod*, *Periodicity* and *Occurrences* attributes of an instance of the *TimeSeries* class shall be such that the minimum periodicity multiplied by the number of occurrences is less than or equal to the duration of the survey period.

7.1.2.8.3 SWE Common encoding

The following table and XML snippet describe how this tasking parameters group shall be encoded using the SWE Common Data Model:

Field Name	Definition	Component
TimeSeries	http://www.opengis.net/def/property/OGC-EO/0/TimeSeries	DataRecord
SurveyPeriod	http://www.opengis.net/def/property/OGC-EO/0/SurveyPeriod The unit SHALL be 'http://www.opengis.net/def/uom/ISO-8601/0/Gregorian' to indicate that the time will have to be encoded using ISO 8601 units and format.	TimeRange
Occurences	http://www.opengis.net/def/property/OGC-EO/0/Occurences	Count
PeriodicityRange	http://www.opengis.net/def/property/OGC-EO/0/PeriodicityRange	QuantityRange
LatestStart	http://www.opengis.net/def/property/OGC-EO/0/LatestStart The unit SHALL be 'http://www.opengis.net/def/uom/ISO-8601/0/Gregorian' to indicate that the time will have to be encoded using ISO 8601 units and format.	Time

```
<swe:item name="TimeSeries">
  <swe:DataRecord definition="http://www.opengis.net/def/property/OGC-EO/0/TimeSeries">
    <swe:field name="SurveyPeriod">
      <swe:TimeRange definition="http://www.opengis.net/def/property/OGC-EO/0/SurveyPeriod">
        <gml:name>Survey Period</gml:name>
        <swe:uom xlink:href="http://www.opengis.net/def/uom/ISO-8601/0/Gregorian"/>
      </swe:TimeRange>
    </swe:field>
    <swe:field name="Occurences">
      <swe:Count definition="http://www.opengis.net/def/property/OGC-EO/0/Occurences">
        <gml:name>Number of Occurences</gml:name>
      </swe:Count>
    </swe:field>
    <swe:field name="PeriodicityRange">
      <swe:QuantityRange definition="http://www.opengis.net/def/property/OGC-EO/0/PeriodicityRange">
        <gml:name>Acquisition Periodicity Range</gml:name>
        <swe:uom code="d"/>
      </swe:QuantityRange>
```

```
    </swe:field>
    <swe:field name="LatestStart">
      <swe:Time definition="http://www.opengis.net/def/property/OGC-EO/0/LatestStart">
        <gml:name>Latest Start</gml:name>
        <swe:uom xlink:href="http://www.opengis.net/def/uom/ISO-8601/0/Gregorian"/>
      </swe:Time>
    </swe:field>
  </swe:DataRecord>
</swe:item>
```

7.1.2.9 AcquisitionType Class

7.1.2.9.1 Description

This parameter is used to specify if the acquisition is monoscopic or stereoscopic, as well as other acquisition parameters.

7.1.2.9.2 Data model

The *AcquisitionType* class is a union of the *MonoscopicAcquisition* and *StereoscopicAcquisition* classes as shown on the UML diagram below:

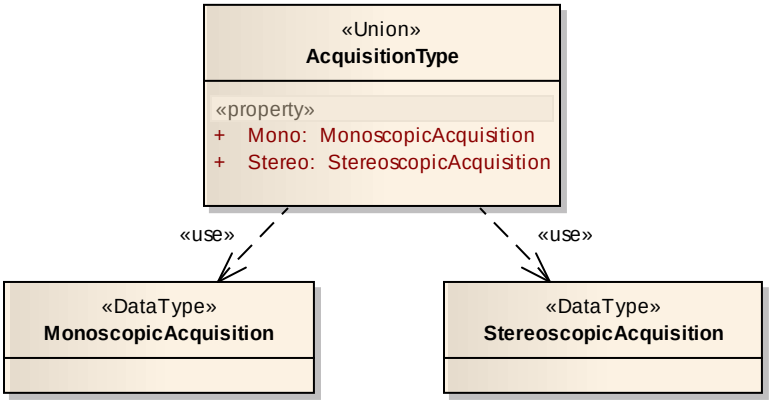

Figure 8 – UML diagram of the *AcquisitionType* class

The *MonoscopicAcquisition* class is specified in clause §7.1.2.10.

The *StereoscopicAcquisition* class is specified in clause §7.1.2.11.

7.1.2.9.3 SWE Common encoding

The following table and XML snippet describe how this tasking parameters group shall be encoded using the SWE Common Data Model:

Field Name	Definition	Component
AcquisitionType	http://www.opengis.net/def/property/OGC-EO/0/AcquisitionType	DataChoice

The following example shows the skeleton of *AcquisitionType* parameter definition (the content of *MonoscopicAcquisition* and *StereoscopicAcquisition* elements has been hidden for clarity):

```
<swe:field name="AcquisitionType">
  <swe:DataChoice definition="http://www.opengis.net/def/property/OGC-EO/0/AcquisitionType">
    <swe:item name="MonoscopicAcquisition" ... />
    <swe:item name="StereoscopicAcquisition" ... />
  </swe:DataChoice>
</swe:field>
```

If only one of monoscopic or stereoscopic acquisition types is supported by a given ground segment, the *DataChoice* shall be replaced by a *DataRecord* with a single field:

```
<swe:field name="AcquisitionType">
  <swe:DataRecord definition="http://www.opengis.net/def/property/OGC-EO/0/AcquisitionType">
    <swe:field name="MonoscopicAcquisition" ... />
  </swe:DataRecord>
</swe:field>
```

7.1.2.10 MonoscopicAcquisition Class

7.1.2.10.1 Description

This group of parameters is used to provide geometrical characteristics of a monoscopic acquisition. In particular, the look angles (incidence or pointing angles) are provided by this class.

7.1.2.10.2 Data model

The *MonoscopicAcquisition* class consists of the attributes listed in the UML diagram below:

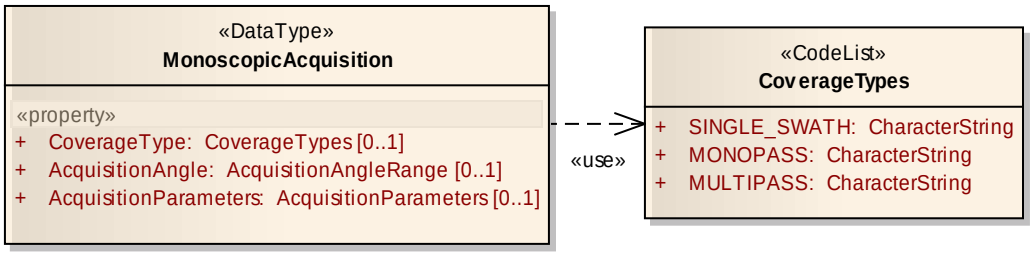

Figure 9 – UML diagram of the *MonoscopicAcquisition* class

Name	Description	Type
CoverageType	Specifies if the imagery should be acquired in one or multiple passes. SINGLE_SWATH: The region of interest should be covered by a single segment. MONOPASS: The region of interest must be covered by one or more segments acquired from the same orbit (some agile satellites can cover large areas even when satisfying this constraint). MULTIPASS: The region of interest can be covered by using images extracted from several segments acquired at different dates provided that they are all acquired within the requested time period.	CodeList SINGLE_SWATH, MONOPASS, MULTIPASS
AcquisitionAngle	Specifies the range of acceptable angles for the acquisition. See section §7.1.2.12 for more details.	
AcquisitionParameters	Specifies additional requested parameters such as ground resolution, instrument modes, etc. See section §7.1.2.15 for more details.	

7.1.2.10.3 SWE Common encoding

The following table and XML snippet describe how this tasking parameters group shall be encoded using the SWE Common Data Model. The *AcquisitionAngle* and *AcquisitionParameters* attributes are not listed in this table as they are detailed in sections §7.1.2.12 and §7.1.2.15 respectively:

Field Name	Definition	Component
Monoscopic Acquisition	http://www.opengis.net/def/property/OGC-EO/0/MonoscopicAcquisition	DataRecord
CoverageType	http://www.opengis.net/def/property/OGC-EO/0/CoverageType Code space shall be "http://www.opengis.net/def/property/OGC-EO/0/CoverageTypes"	Category

Contents of the *AcquisitionAngle* and *AcquisitionParameters* "field" elements are hidden in the XML snippet below for clarity:

```
<swe:field name="MonoscopicAcquisition">
  <swe:DataRecord definition="http://www.opengis.net/def/property/OGC-EO/0/MonoscopicAcquisition">
    <swe:field name="CoverageType">
      <swe:Category definition="http://www.opengis.net/def/property/OGC-EO/0/CoverageType">
        <gml:name>Coverage Type</gml:name>
        <swe:codeSpace xlink:href="http://www.opengis.net/def/property/OGC-EO/0/CoverageTypes"/>
        <swe:constraint>
          <swe:AllowedTokens>
            <swe:value>SINGLE_SWATH</swe:value>
            <swe:value>MONOPASS</swe:value>
            <swe:value>MULTIPASS</swe:value>
          </swe:AllowedTokens>
        </swe:constraint>
        <swe:value>MULTIPASS</swe:value>
      </swe:Category>
```



```
    </swe:field>
    <swe:field name="AcquisitionAngle" ... />
    <swe:field name="AcquisitionParameters" ... />
  </swe:DataRecord>
</swe:field>
```

7.1.2.11 StereoscopicAcquisition Class

7.1.2.11.1 Description

This group of parameters is used to provide geometrical characteristics of a stereoscopic acquisition. In particular, the look angles (incidence or pointing angles) for forward and rear views are provided by this class.

7.1.2.11.2 Data model

The *StereoscopicAcquisition* class consists of the attributes listed in the UML diagram below:

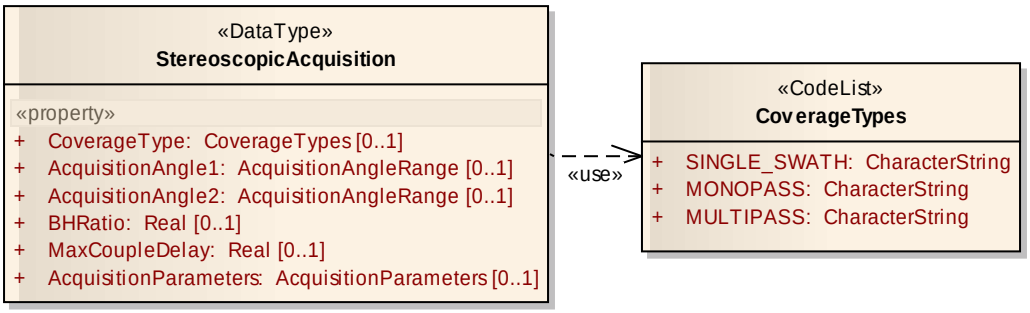

Figure 10 – UML diagram of the *StereoscopicAcquisition* class

Name	Description	Type (unit)
CoverageType	See section §7.1.2.10.2 for the description of this parameter.	CodeList SINGLE_SWATH, MONOPASS, MULTIPASS
AcquisitionAngle1	Specifies the acceptable range of viewing angles for the forward view. See §7.1.2.12 for more details.	AcquisitionAngleRange
AcquisitionAngle2	Specifies the acceptable range of acquisition angles for the rear view. See §7.1.2.12 for more details.	AcquisitionAngleRange
BHRatio	Ratio between base and height of the triangle constructed by viewing directions of the two acquisitions in a stereoscopic couple, and the local horizontal plane.	Real (unitless)
MaxCoupleDelay	Maximum time delay between two acquisitions of a stereoscopic couple.	Real (days)
AcquisitionParameters	Specifies additional requested parameters such as ground resolution, instrument modes, etc. See §7.1.2.15 for details.	AcquisitionParameters

7.1.2.11.3 SWE Common encoding

The following table and XML snippet describe how this tasking parameters group shall be encoded using the SWE Common Data Model. The *CoverageType*, *AcquisitionAngle* and *AcquisitionParameters* attributes are not listed in this table as they are detailed in sections §7.1.2.10, §7.1.2.12 and §7.1.2.15 respectively:

Field Name	Definition	Component
Stereoscopic Acquisition	http://www.opengis.net/def/property/OGC-EO/0/StereoscopicAcquisition	DataRecord
BHRatio	http://www.opengis.net/def/property/OGC-EO/0/BHRatio	Quantity
MaxCoupleDelay	http://www.opengis.net/def/property/OGC-EO/0/MaxCoupleDelay	Quantity

```
<swe:field name="StereoscopicAcquisition">
  <swe:DataRecord definition="http://www.opengis.net/def/property/OGC-EO/0/StereoscopicAcquisition">
    <swe:field name="CoverageType" ... />
    <swe:field name="AcquisitionAngle1" ... />
    <swe:field name="AcquisitionAngle2" ... />
    <swe:field name="BHRatio">
      <swe:Quantity definition="http://www.opengis.net/def/property/OGC-EO/0/BHRatio">
        <gml:name>B/H Ratio</gml:name>
        <swe:uom code="1"/>
        <swe:constraint>
          <swe:AllowedValues>
            <swe:interval>0.35 3.4</swe:interval>
          </swe:AllowedValues>
        </swe:constraint>
        <swe:value>30</swe:value>
      </swe:Quantity>
    </swe:field>
    <swe:field name="MaxCoupleDelay">
      <swe:Quantity definition="http://www.opengis.net/def/property/OGC-EO/0/MaxCoupleDelay">
        <gml:name>Maximum Couple Delay</gml:name>
        <swe:uom code="d"/>
        <swe:constraint>
          <swe:AllowedValues>
            <swe:interval>0 +Inf</swe:interval>
          </swe:AllowedValues>
        </swe:constraint>
        <swe:value>30</swe:value>
      </swe:Quantity>
    </swe:field>
    <swe:field name="AcquisitionParameters" ... />
  </swe:DataRecord>
</swe:field>
```

Contents of the “constraint” and “value” elements are not normative. They can be redefined by each EO-SPS instance or even completely omitted.

7.1.2.12 AcquisitionAngleRange Class

7.1.2.12.1 Description

This parameter group allows the specification of specific angles for an acquisition in the case the satellite or instrument is steerable along one or more axes.

7.1.2.12.2 Data model

This abstract class has two concrete sub-classes defined in this specification.

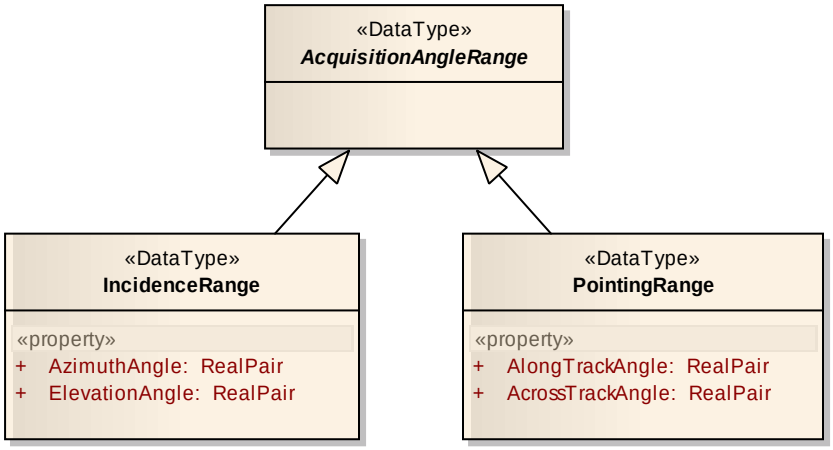

Figure 11 – UML diagram of the *AcquisitionAngleRange* class and its sub-classes

This model allows acceptable acquisition angles to be specified either by indicating a range of incidence angles or pointing angles.

The *IncidenceRange* class is specified in clause §7.1.2.13.

The *PointingRange* class is specified in clause §7.1.2.14.

7.1.2.13 IncidenceRange Class

7.1.2.13.1 Description

Specification of allowed acquisition angles in terms of angles expressed relative to a frame of reference attached to the location of the acquisition on the ground. More precisely the angle of incidence is the angle made by the look vector with the local vertical direction (i.e. normal to the earth ellipsoid where the look vector intersects the earth surface).

7.1.2.13.2 Data model

Name	Description	Type (unit)
AzimuthAngle	Range of acceptable azimuth incidence angles. The azimuth angle is the angle that the look vector (projected vertically on the earth surface) makes with the north direction. It thus indicates from which geographic direction (i.e. North, East, West, South) the region of interest should be imaged.	RealPair (degrees)
ElevationAngle	Range of acceptable elevation incidence angles. The elevation angle is the angle that the look vector makes with the local vertical. It thus indicates how vertically the region should be imaged. This is the traditional meaning of incidence angle.	RealPair (degrees)

7.1.2.13.3 SWE Common encoding

The following table and XML snippet describe how this tasking parameters group shall be encoded using the SWE Common Data Model.

Field Name	Definition	Component
AcquisitionAngle	http://www.opengis.net/def/property/OGC-EO/0/IncidenceRange	DataRecord
AzimuthAngle	http://www.opengis.net/def/property/OGC-EO/0/AzimuthAngle	QuantityRange
ElevationAngle	http://www.opengis.net/def/property/OGC-EO/0/ElevationAngle	QuantityRange

```
<swe:field name="AcquisitionAngle">
  <swe:DataRecord definition="http://www.opengis.net/def/property/OGC-EO/0/IncidenceRange">
    <swe:field name="AzimuthAngle">
      <swe:QuantityRange definition="http://www.opengis.net/def/property/OGC-EO/0/AzimuthAngle">
        <gml:name>Azimuth Range</gml:name>
        <swe:uom code="deg"/>
        <swe:constraint>
          <swe:AllowedValues>
            <swe:interval>-180 +180</swe:interval>
          </swe:AllowedValues>
        </swe:constraint>
        <swe:value>-180 +180</swe:value>
      </swe:QuantityRange>
    </swe:field>
    <swe:field name="ElevationAngle">
      <swe:QuantityRange definition="http://www.opengis.net/def/property/OGC-EO/0/ElevationAngle">
        <gml:name>Elevation Range</gml:name>
        <swe:uom code="deg"/>
        <swe:constraint>
          <swe:AllowedValues>
            <swe:interval>0 31</swe:interval>
          </swe:AllowedValues>
        </swe:constraint>
        <swe:value>0 31</swe:value>
      </swe:QuantityRange>
    </swe:field>
  </swe:DataRecord>
</swe:field>
```


Contents of the "*swe:constraint*" and "*swe:value*" elements are not normative. They can be redefined by each EO-SPS instance or even completely omitted.

7.1.2.14 PointingRange Class

7.1.2.14.1 Description

Specification of allowed acquisition angles in terms of angles expressed in the satellite reference frame and relative to the nadir direction. More precisely, the pointing angle is the angle made by the look vector with the nadir direction. The nadir direction is the vector pointing down from the satellite center of mass and intersecting the earth ellipsoid in a way that it is orthogonal to the local horizontal plane at the location of the intersection.

7.1.2.14.2 Data model

Name	Description	Type (unit)
AlongTrackAngle	Range of acceptable pointing angles in the along track direction (i.e. the angle between the nadir direction and the look vector, measured around the axis of the satellite reference frame that is orthogonal to the orbit plane).	RealPair (degrees)
AcrossTrackAngle	Range of acceptable pointing angles in the across track direction (i.e. the angle between the nadir direction and the look vector, measured around the axis of the satellite reference frame with a direction tangent to the satellite trajectory).	RealPair (degrees)

7.1.2.14.3 SWE Common encoding

The following table and XML snippet describe how this tasking parameters group shall be encoded using the SWE Common Data Model.

Field Name	Definition	Component
AcquisitionAngle	http://www.opengis.net/def/property/OGC-EO/0/PointingRange	DataRecord
AlongTrackAngle	http://www.opengis.net/def/property/OGC-EO/0/AlongTrackAngle	QuantityRange
AcrossTrackAngle	http://www.opengis.net/def/property/OGC-EO/0/AcrossTrackAngle	QuantityRange

```
<swe:field name="AcquisitionAngle">
  <swe:DataRecord definition="http://www.opengis.net/def/property/OGC-EO/0/PointingRange">
    <swe:field name="AlongTrackAngle">
      <swe:QuantityRange definition="http://www.opengis.net/def/property/OGC-EO/0/AlongTrackAngle">
        <gml:name>Along-Track Pointing Range</gml:name>
        <swe:uom code="deg"/>
        <swe:constraint>
          <swe:AllowedValues>
            <swe:interval>-60 +60</swe:interval>
          </swe:AllowedValues>
        </swe:constraint>
        <swe:value>-30 +60</swe:value>
```

```
      </swe:QuantityRange>
    </swe:field>
    <swe:field name="AcrossTrackAngle">
      <swe:QuantityRange definition="http://www.opengis.net/def/property/OGC-EO/0/AcrossTrackAngle">
        <gml:name>Across-Track Pointing Range</gml:name>
        <swe:uom code="deg"/>
        <swe:constraint>
          <swe:AllowedValues>
            <swe:interval>-60 +60</swe:interval>
          </swe:AllowedValues>
        </swe:constraint>
        <swe:value>-30 +30</swe:value>
      </swe:QuantityRange>
    </swe:field>
  </swe:DataRecord>
</swe:field>
```

Contents of the "*swe:constraint*" and "*swe:value*" elements are not normative. They can be redefined by each EO-SPS instance or even completely omitted.

7.1.2.15 AcquisitionParameters Class

7.1.2.15.1 Description

This group contains tasking parameters related to the instrument configuration to use for the acquisition. These parameters are often expressed in terms of an acquisition mode or directly by the desired ground resolution.

7.1.2.15.2 Data model

This class is represented in the core set of tasking parameters as an abstract class since acquisition parameters are specific to the type of mission.

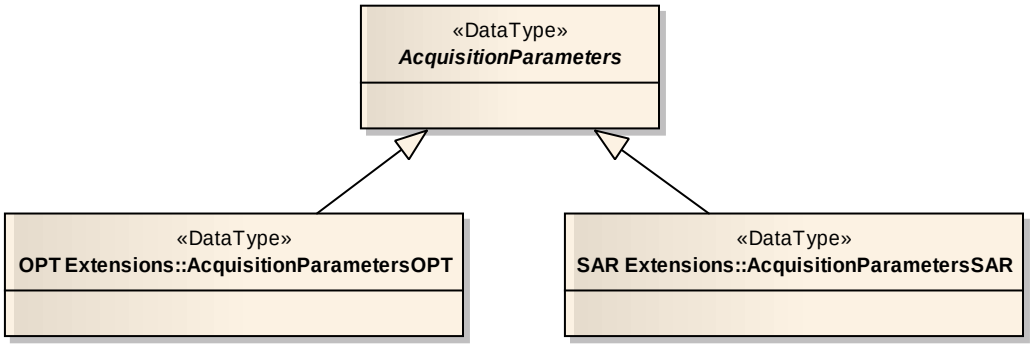

Figure 12 – UML diagram of the *AcquisitionParameters* class

Concrete implementations of this class are defined in separate sections of this standard that define additional requirements for the support of optical and radar earth observation missions.

The *AcquisitionParametersOPT* class is the concrete implementation for optical instruments and is specified in §7.2.2.

The *AcquisitionParametersSAR* class is the concrete implementation for synthetic aperture radar (SAR) instruments and is specified in §7.3.2.

7.1.2.16 ValidationParameters Class

7.1.2.16.1 Description

This group contains tasking parameters related to the test to be done before an acquisition is validated after it has been acquired.

7.1.2.16.2 Data model

This class is represented in the core set of tasking parameters as an abstract class since validation parameters are specific to the type of mission.

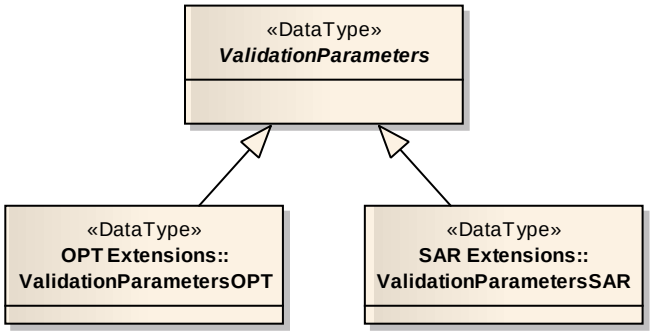

Figure 13 – UML diagram of the *ValidationParameters* class

Concrete implementations of this class are defined in separate sections of this standard that define additional requirements for the support of optical and radar earth observation missions.

The *ValidationParametersOPT* class is the concrete implementation for optical instruments and is specified in §7.2.2.2.

The *ValidationParametersSAR* class is the concrete implementation for synthetic aperture radar (SAR) instruments and is specified in §7.3.2.2.

7.1.3 Auxiliary parameters

The parameters defined in this clause are not considered tasking parameters because they are not related to the tasking itself. Additionally they are not always applicable to both *GetFeasibility* and *Submit* requests and thus their definition cannot be retrieved with the

DescribeTasking operation. They are instead defined as XML elements that shall be inserted in extension slots of the SPS schema.

7.1.3.1 FeasibilityLevel Element

7.1.3.1.1 Description

The feasibility level is an auxiliary parameter used to specify the required level of precision of a feasibility study. This parameter is added to regular SPS *GetFeasibility* requests and can take two possible values: 'SIMPLE' and 'COMPLETE'.

The 'SIMPLE' mode results in a rough simulation of the feasibility, while the 'COMPLETE' mode indicates that the server should generate the best possible feasibility study (i.e. as close to the reality as possible). This means that a 'SIMPLE' feasibility algorithm may not take all parameters into account (for instance, the actual workload or the climate forecast may be ignored) while the 'COMPLETE' version will take as much information as possible into account. The exact meaning of these two levels is however up to each data provider and should be expressed in the terms of service. The parameters taken into account in both modes shall be indicated in the *GetFeasibilityResponse* using the *informationUsed* attribute of the *FeasibilityStudy* class (see 7.1.4.2).

One important common point between missions is that a 'COMPLETE' feasibility study will usually take longer (and eventually involve human actions) than a 'SIMPLE' one (which can often be entirely done automatically). Furthermore, a request for a 'COMPLETE' feasibility study may require stronger authentication and possibly the payment of a fee. This parameter is only applicable to SPS *GetFeasibility* requests. The feasibility level is an optional parameter of SPS *GetFeasibility* requests. When it is omitted, the server shall assume that a 'SIMPLE' feasibility is requested.

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/core/feasibility-level-default
Req 10. The *FeasibilityLevel* parameter shall be taken into account when it is inserted within a *GetFeasibility* request. When it is omitted the default value of 'SIMPLE' shall be assumed by the server.

7.1.3.1.2 XML encoding

The following schema snippet shows the global element declaration that normalizes the encoding of this parameter in XML:

```
<element name="FeasibilityLevel" type="eosps:FeasibilityLevelType"/>

<simpleType name="FeasibilityLevelType">
  <union memberTypes="eosps:FeasibilityLevelEnumerationType eosps:FeasibilityLevelOtherType"/>
</simpleType>
```



```
<simpleType name="FeasibilityLevelEnumerationType">
  <restriction base="string">
    <enumeration value="SIMPLE"/>
    <enumeration value="COMPLETE"/>
  </restriction>
</simpleType>

<simpleType name="FeasibilityLevelOtherType">
  <restriction base="string">
    <pattern value="other: \w{2,}"/>
  </restriction>
</simpleType>
```

This parameter shall be encapsulated in an extension slot of a *GetFeasibility* request as show below:

```
<sps:GetFeasibility
   xmlns:sps="http://www.opengis.net/sps/2.0"
   xmlns:swes="http://www.opengis.net/swes/2.0"
   xmlns:eosps="http://www.opengis.net/eosps/2.0">
  <swes:extension>
    <eosps:FeasibilityLevel>COMPLETE</eosps:FeasibilityLevel>
  </swes:extension>
    ...
</sps:GetFeasibility>
```

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/core/feasibility-level-valid
Req 11. The *FeasibilityLevel* XML element shall be valid with respect to the *spsRequestExtensions.xsd* schema and inserted within an extension element of the *GetFeasibility* request.

7.1.3.2 ReferenceFeasibilityID Element

7.1.3.2.1 Description

The *ReferenceFeasibilityID* tag can be inserted within *Submit* or *Reserve* requests to reference a previously generated feasibility study. The *ReferenceFeasibilityID* tag can be omitted by clients but servers shall read and store its value for the same amount of time as the feasibility study or task information itself.

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/core/ref-feasibility-id-used
Req 12. A *Submit* or *Reserve* request containing the *ReferenceFeasibilityID* parameter shall be associated with the corresponding feasibility study.

After a feasibility study has been requested and approved by a client, a task with identical parameters can be submitted (or reserved), provided the referred feasibility study is not yet expired (as indicated by its expiration date. See section 7.1.4.2). The provider can

then make sure acquisitions are executed as closely as possible to what was agreed upon in the feasibility assessment.

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/core/ref-feasibility-id-correct
Req 13. The value of *ReferenceFeasibilityID* parameter shall be the identifier of an existing and active (i.e. not expired) feasibility study report.

For such a request to be valid, the tasking parameters values specified in the *Submit* operation shall match the ones used in the previous *GetFeasibility* request.

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/core/ref-feasibility-id-params
Req 14. The values of tasking parameters included in a *Submit* or *Reserve* request shall be identical to the ones used in the *GetFeasibility* request referred to by the *ReferenceFeasibilityID* parameter.

This feature is especially useful because certain data providers consider the acceptance of a 'complete' feasibility study by the customer as contractually binding for both parties. The acceptance of the feasibility study is triggered by the submission of the task (via the *Submit* operation) with a *ReferenceFeasibilityID* referring to a previously generated feasibility study. Note that each data provider is responsible for specifying the exact consequences of this acceptance in its terms of services. In particular, data providers can consider feasibility studies as informational only.

Note that the only way for a client to "accept" a feasibility study is to use the *ReferenceFeasibilityID* tag in a subsequent *Submit* or *Reserve* request. A tasking request with no such tag will be processed completely separately, and in this case, the data provider is free to satisfy the request in any way desired without disclosing how it will be done to the client.

7.1.3.2.2 XML encoding

The following schema snippet shows the global element declaration that normalizes the encoding of this parameter in XML:

```
<element name="ReferenceFeasibilityID" type="string"/>
```

This parameter shall be encapsulated in an extension slot of a *Submit* or *Reserve* request as show below:

```
<sps:Submit
  xmlns:sps="http://www.opengis.net/sps/2.0"
  xmlns:swes="http://www.opengis.net/swes/2.0"
  xmlns:eosps="http://www.opengis.net/eosps/2.0">
```



```
  <swes:extension>
    <eosps:ReferenceFeasibilityID>
       http://ws.spotimage.com/sps/feasibility/F2AEA58FFA12B56CC
    </eosps:ReferenceFeasibilityID>
  </swes:extension>
    ...
</sps:Submit>
```

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/core/ref-feasibility-id-valid
Req 15. The *ReferenceFeasibilityID* XML element shall be valid with respect to the *spsRequestExtensions.xsd* schema and inserted within an extension element of the *Submit* or *Reserve* XML request.

7.1.4 Feasibility study model

7.1.4.1 Introduction

In addition to the basic information included in the feasibility response defined in the SPS standard, detailed feasibility study results required for proper operations of EO satellites shall also be provided by an SPS supporting this extension. This information shall be inserted in status reports produced as the result of a feasibility request and retrieved either directly with the *GetFeasibility* request or via a secondary *GetStatus* request. The *FeasibilityStudy* class described in this clause describes the information model to be used to provide detailed feasibility study results.

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/core/feasibility-study-report
Req 16. A *StatusReport* describing the result of a feasibility study and with a request status of '*Accepted*' shall include an instance of the *FeasibilityStudy* class.

Note: Feasibility results can also be provided for information when the demand is not feasible (i.e. the request status in the response to GetFeasibility is "Rejected"). In this case, the response will generally contain an incomplete list of segments (i.e. a portion of the region of interest is not covered by any of the segments in the list) and/or a list of cells with at least some of them having a very low probability of success.

7.1.4.2 FeasibilityStudy Class

7.1.4.2.1 Description

The results of a feasibility study are composed of an overall feasibility assessment, plus a detailed feasibility, per cell or per segment, depending on the mission.

For some EO systems, feasible acquisition segments can be estimated long in advance and be provided directly in the results. For each planned segment, several properties are then specified such as the date and time of acquisition, the combination of platform, instrument and mode used to acquire the segment, acquisition angles and ground footprint of the segment, etc... The customer can then see the exact segment(s) that will be acquired for him.

For other EO systems, such as highly agile satellites, feasible segments are not known in advance due to the possibility of acquiring multiple areas of the earth at a given instant in time (i.e. by pointing the satellite). In such cases, a frequent re-optimization of the satellite work plan with respect to the latest weather conditions (especially for optical instruments) or other criteria is often done in order to satisfy a maximum number of customers. The consequence is that only an estimated list of potential segments can be generated with no certainty that these segments will actually be acquired exactly as planned.

One common method to solve the problem of work plan optimization for agile satellites is the division of the region of interest in several smaller sub-areas called grid cells that are processed individually in a statistical manner. Since the actual chosen/successful segment covering each cell is not known with certainty at the time of the feasibility study, statistical information is instead given to indicate the chance of successfully covering each cell, as well as the overall chance of successfully covering the whole area before the end of the requested time window. Note that these probabilities can take into account climate, weather and satellite workload conditions, as indicated in the feasibility study.

In such a case, the feasibility result contains a list of cells in addition to the list of planned segments (planned segments are optional if cells are given). Each cell feasibility assessment can also include information such as the estimated date of success and the date of the first possible attempt. For advanced users, it is also possible to include all possible acquisition attempts for each cell (instead of a single estimated segment per cell) in the feasibility response.

7.1.4.2.2 Data model

The feasibility study is a top level class and aggregates either a list of segments, a list of cells or both (as explained above). It shall be inserted directly in the extension slot of a *StatusReport* element.

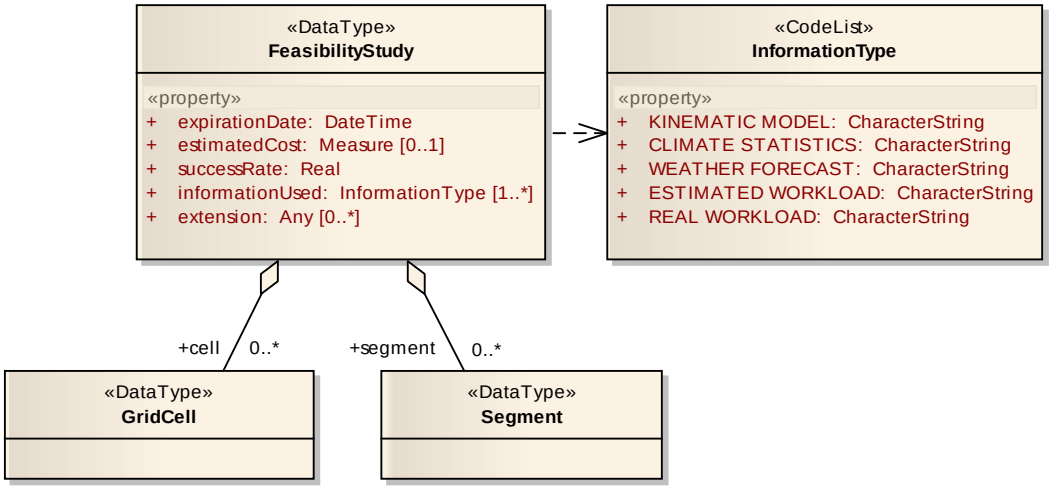

Figure 14 – UML diagram of the *FeasibilityStudy* class

Attributes of this class are described below:

Name	Description	Type (unit)
expirationDate	Date of expiration of the feasibility study (i.e. the task is guaranteed to be executed as described in the feasibility study if submitted before this date).	DateTime
estimatedCost	Estimated cost to cover the requested area with the requested parameters. This includes the cost of the programming and the cost of the acquired image themselves.	Measure (currency unit)
successRate	Overall success rate of the proposed acquisition. This is the probability that the whole requested area will be covered (possibly N times in the case of time series) with the specified acquisition parameters before the end of the time period.	Real (%)
informationUsed	Type of information used to do the simulation and compute the feasibility result. Several ones can be specified.	CodeList - KINEMATIC MODEL - CLIMATE STATISTICS - WEATHER FORECAST - ESTIMATED WORKLOAD - REAL WORKLOAD
extension	Extension slot for mission specific information or future extensions to this standard.	Any
cell	A list of cells is included when providing feasibility results in a statistical form. Each cell represents a subset of the region of interest for which statistics have been computed. Cells should not overlap.	GridCell
segment	List of potential segments to be acquired to cover the region of interest. Note that when the same area can be covered by different segments at different dates, all the possible segments can be listed here but only some of them will actually be acquired. Segments can overlap.	Segment

The *GridCell* and *Segment* classes are specified in §7.1.6.1 and §7.1.6.2 respectively.

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/core/feasibility-study-acc-obj-status
Req 17. The value of the *status* attribute of *GridCell* and *Segment* instances shall be 'POTENTIAL' when these objects are used within a feasibility study report with a request status of 'Accepted' (i.e. feasible).

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/core/feasibility-study-rej-obj-status
Req 18. The value of the *status* attribute of the *GridCell* and *Segment* instances shall be 'POTENTIAL' or 'REJECTED' when these objects are used within a feasibility study report with a request status of 'Rejected' (i.e. not feasible).

7.1.4.2.3 XML encoding

The following schema snippet normalizes the structure of the *FeasibilityStudy* XML element:

```
<element name="FeasibilityStudy" type="eosps:FeasibilityStudyType" substitutionGroup="gml:AbstractObject"/>

<complexType name="FeasibilityStudyType">
  <sequence>
    <element name="expirationDate" type="dateTime"/>
    <element name="estimatedCost" type="gml:MeasureType" minOccurs="0"/>
    <element name="successRate" type="double"/>
    <element name="informationUsed" type="eosps:InformationTypeType" maxOccurs="unbounded"/>
    <element name="extension" type="anyType" minOccurs="0" maxOccurs="unbounded"/>
    <element name="segment" type="eosps:SegmentPropertyType" minOccurs="0" maxOccurs="unbounded"/>
    <element name="cell" type="eosps:GridCellPropertyType" minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
  <attribute ref="gml:id" use="optional"/>
</complexType>
```

This element shall be inserted within an extension slot of the *StatusReport* element defined in the SPS interface standard [OGC 09-000]. The example below shows this within a *GetFeasibilityResponse* element:

```
<sps:GetFeasibilityResponse
  xmlns:sps="http://www.opengis.net/sps/2.0"
  xmlns:swes="http://www.opengis.net/swes/2.0"
  xmlns:eosps="http://www.opengis.net/eosps/2.0">
  <sps:result>
    <sps:StatusReport>
      <swes:extension>
        <eosps:FeasibilitysStudy>
          <eosps:expirationDate>2010-05-27T10:30:32Z</eosps:expirationDate>
          <eosps:successRate>82</eosps:successRate>
          <eosps:informationUsed>KINEMATIC MODEL</eosps:informationUsed>
          <eosps:informationUsed>CLIMATE STATISTICS</eosps:informationUsed>
          <eosps:segment ... />
          <eosps:cell ... />
        </eosps:FeasibilitysStudy>
```

```
      </swes:extension>
      <sps:task>http://ws.spotimage.com/sps/tasks/F0112F56ADE2A56CB</sps:task>
      <sps:procedure>http://ws.spotimage.com/sps/sensors/SPOT-Constellation</sps:procedure>
      <sps:status>Accepted</sps:status>
      <sps:updateTime>2010-05-20T10:30:32Z</sps:updateTime>
    </sps:StatusReport>
  </sps:result>
</sps:GetFeasibilityResponse>
```

Feasibility study information can be inserted within a *GetFeasbilityResponse* element or within a *GetStatusResponse* element in the case the call to *GetStatus* was used to retrieve the results of a feasibility study asynchronously.

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/core/feasibility-study-valid
Req 19. The *FeasibilityStudy* XML element shall be inserted in the extension slot of the SPS *StatusReport* element describing the status of a feasibility study and be valid with respect to the *eoTaskingExtensions.xsd* schema.

Details of the *segment* and *cell* elements have been omitted for clarity in the snippet above and are detailed in sections §0 and §7.1.6.2 respectively. Full examples of *StatusReport* containing feasibility analysis results are provided in Annex C.

7.1.5 Programming status model

7.1.5.1 Introduction

In addition to the basic information included in the status reports defined in the SPS standard, detailed EO system programming status shall also be provided by an SPS supporting this extension. Such information is essential to provide a spatio-temporal view of the advancement of on-going data acquisitions. This information shall be inserted in the *StatusReport* (in the response to a *GetStatus*, *Submit*, *Update* or *Confirm* request) resulting from a task submission as soon as the task is in execution (i.e. task status changes to "*InExecution*"). The *ProgrammingStatus* class described in this clause describes the information model and encoding to be used to provide detailed status of on-going acquisitions.

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/core/prog-status-report
Req 20. A *StatusReport* describing the state of a programming task and with the task status 'InExecution' shall include an instance of the *ProgrammingStatus* class.

Note: Programming status can also be provided for information when the demand is rejected. In this case, the response can contain a list of cells and/or segments indicating which ones have been rejected.

7.1.5.2 ProgrammingStatus Class

7.1.5.2.1 Description

This class allows the programming service to describe the status of EO acquisitions in details, enabling the user to follow the advancement of his tasks both temporally and spatially.

As described in section §7.1.4, this SPS extension is designed to satisfy the needs of different types of EO systems and thus allows reporting status for both cells and segments. All acquired and validated segments shall be included in the list. Segments with different states (i.e. potential, planned, cancelled, rejected or failed) can optionally be included for information.

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/core/prog-status-inc-segments
Req 21. An instance of the *ProgrammingStatus* class shall include all segments that have been acquired (i.e. in the 'ACQUIRED' or 'VALIDATED' state) at the time the report is generated.

If the programming system generates feasibility results using a discretization in cells, the status of these cells shall also be included in the programming status class.

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/core/prog-status-inc-cells
Req 22. An instance of the *ProgrammingStatus* class shall include the list of all cells resulting from the corresponding feasibility study (if any), with an updated status.

7.1.5.2.2 Data model

The *ProgrammingStatus* class is a top level class and aggregates either a list of segments, a list of cells or both (as required above). It shall be inserted directly in the extension slot of a *StatusReport* element.

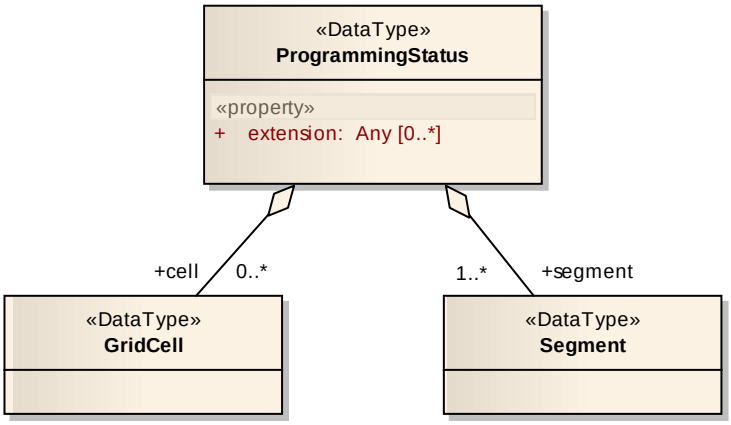

Figure 15 – UML diagram of the *ProgrammingStatus* class

As shown on the diagram, a list of segments shall always be provided while the list of cells is optional in general. However, for the sake of consistency, the list of cells status is mandatory if the service uses cells as a way to express feasibility results (see §7.1.4).

The following table describes attributes of this class.

Name	Description	Type (unit)
extension	Extension slot for mission specific information or future extensions to this standard.	Any
cell	List of cells corresponding to the cells listed in the feasibility study. Cells status information shall be updated every time a new acquisition is made.	GridCell
segment	List of segments that have been or are going to be acquired to cover the requested area. The *ProgrammingStatus* class shall at least contain all segments with a status of “ACQUIRED” or “VALIDATED” but can also contain potential, planned, cancelled, rejected and failed segments.	Segment

The *GridCell* and *Segment* classes are specified in §7.1.6.1 and §7.1.6.2 respectively.

7.1.5.2.3 XML encoding

The following schema snippet normalizes the structure of the *ProgrammingStatus* XML element:

```
<element name="ProgrammingStatus" type="eosps:ProgrammingStatusType"
substitutionGroup="gml:AbstractObject"/>

<complexType name="ProgrammingStatusType">
  <sequence>
    <element name="extension" type="anyType" minOccurs="0" maxOccurs="unbounded"/>
    <element name="segment" type="eosps:SegmentPropertyType" maxOccurs="unbounded"/>
    <element name="cell" type="eosps:GridCellPropertyType" minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
```

```
  <attribute ref="gml:id" use="optional"/>
</complexType>
```

This element shall be inserted within an extension slot of the *StatusReport* element defined in the SPS interface standard [OGC 09-000]. The example below shows this within a *GetStatusResponse* element:

```
<sps:GetStatusResponse
  xmlns:sps="http://www.opengis.net/sps/2.0"
  xmlns:swes="http://www.opengis.net/swes/2.0"
  xmlns:eosps="http://www.opengis.net/eosps/2.0">
  <sps:status>
    <sps:StatusReport>
      <swes:extension>
        <eosps:ProgrammingStatus>
          <eosps:segment ... />
          <eosps:segment ... />
          <eosps:cell ... />
        </eosps:ProgrammingStatus>
      </swes:extension>
      <sps:task>http://ws.spotimage.com/sps/feasibility/F2AEA58FFA12B56CC</sps:task>
      <sps:procedure>http://ws.spotimage.com/sps/sensors/SPOT-Constellation</sps:procedure>
      <sps:requestStatus>Accepted</sps:requestStatus>
      <sps:updateTime>2010-06-20T11:16:32Z</sps:updateTime>
    </sps:StatusReport>
  </sps:status>
</sps:GetStatusResponse>
```

Programming status information can be inserted within a *GetStatusResponse* element but also within *SubmitResponse*, *UpdateResponse* or *ConfirmResponse* when the task is accepted synchronously.

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/core/prog-status-valid
Req 23. The *ProgrammingStatus* element shall be inserted in the extension slot of the *StatusReport* element describing the status of a submitted task and be valid with respect to the *eoTaskingExtensions.xsd* schema.

Details of the *segment* and *cell* elements have been omitted for clarity in the snippet above. Full examples of *StatusReport* within *GetStatus* operation responses and containing detailed programming status are provided in Annex C.

7.1.6 GridCell and Segment model

7.1.6.1 GridCell Class

7.1.6.1.1 Description

This class is used to provide parameters of one of the grid cells used to discretize the region of interest. It can be used to provide cell feasibility assessment as well as to provide status of cells coverage.

These cell parameters shall be consistent with the corresponding tasking request. In particular the cell's ground footprint shall intersect the region of interest, the date of the first attempt shall be within the time period requested if the status is "POTENTIAL", etc.

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/core/gridcell-coherent
Req 24. The attribute values of a *GridCell* instance shall be consistent with what was requested in the original feasibility or tasking request.

7.1.6.1.2 Data model

The UML diagram below shows the *GridCell* class that is composed of a list of simple parameters giving the characteristics of each grid cell. Parameters have slightly different meanings when used within feasibility studies than when they are used to provide detailed programming status.

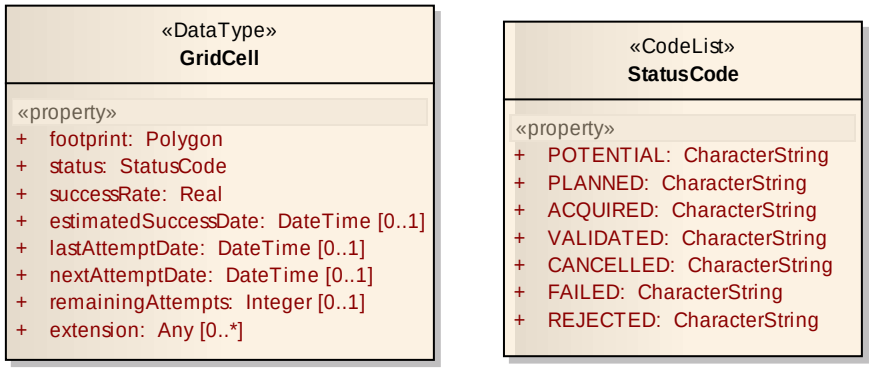

Figure 16 – UML diagram of the *GridCell* class

Name	Description	Type (unit)
footprint	Ground footprint of the cell. The GML polygon shall be expressed in the EPSG 4326 reference system.	Polygon *(OGC 07-036)*
status	State of coverage of this cell. Descriptions of status codes are given in the table below. The value shall be 'POTENTIAL' in the case of a	StatusCode CodeList

	feasibility study, unless the cell cannot be covered at all in which case the status shall be 'REJECTED'.	
successRate	Probability that this cell will be covered by an acceptable image (i.e. depending on validation criteria) before the end of the requested period.	Real (%)
estimatedSuccessDate	The estimated worst acquisition date (i.e. latest date at which the cell should be covered and validated). This value should be re-estimated in case the programming is delayed.	DateTime
lastAttemptDate	The date and time of the last acquisition attempt to cover this cell whether it was successful or not. This shall be omitted in the case of a feasibility study.	DateTime
nextAttemptDate	The date and time of the next acquisition attempt that can possibly be used to cover this cell. In the case of a feasibility study, this can be set to the time of the first possible attempt. It should be updated every time the status changes and usually omitted as soon as the cell state changes to 'VALIDATED'.	DateTime
remainingAttemps	Number of remaining possible acquisition attempts within the requested period that allow covering the cell. In the case of a feasibility study this is the total number of attempts that are possible within the requested period. It is usually omitted as soon as the cell state changes to 'VALIDATED'.	Integer
extension	Extension slot for mission specific information or future extensions to this standard.	Any

The table below describes the meaning of status codes used in the *GridCell* class:

Name	Description
POTENTIAL	Cells are marked as potential when they are estimated in the context of a feasibility study.
PLANNED	At least one segment covering the cell has been planned. This state shall be used when the first attempt to cover the cell is planned or when additional attempts are planned after previous ones have failed (i.e. previous image segments covering the cell did not pass the validation criteria such as cloud cover). The *nextAttemptDate* shall be the forecasted acquisition date of the planned attempt and the *lastAttemptDate* shall be the date of the previous (usually unsuccessful) attempt.
VALIDATED	The last acquired segment covering the cell has been validated. This is usually a final state meaning that an acceptable image has been obtained and that no more acquisition attempts need to be made to cover the cell.
CANCELLED	The cell has been cancelled by the client. No more acquisition attempts will be made to cover it so the *nextAttemptDate* field shall be omitted and the *remainingAttempts* field shall be 0.
REJECTED	The cell has been rejected by the programming system because it cannot be covered at all before the end of the requested period. No more acquisition attempts can possibly cover it so the *nextAttemptDate* field shall be omitted and the *remainingAttempts* field shall be 0.
ACQUIRED	Not used in grid cells.
FAILED	Not used in grid cells.

7.1.6.1.3 XML encoding

The following schema snippet normalizes the structure of the *GridCell* XML element:

```
<element name="GridCell" type="eosps:GridCellType" substitutionGroup="gml:AbstractObject"/>

<complexType name="GridCellType">
  <sequence>
    <element name="footprint" type="gml:SurfacePropertyType"/>
    <element name="status" type="eosps:StatusCodeType"/>
    <element name="successRate" type="double"/>
    <element name="estimatedSuccessDate" type="dateTime" minOccurs="0"/>
    <element name="lastAttemptDate" type="dateTime" minOccurs="0"/>
    <element name="nextAttemptDate" type="dateTime" minOccurs="0"/>
    <element name="remainingAttempts" type="integer" minOccurs="0"/>
    <element name="extension" type="anyType" minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
  <attribute ref="gml:id" use="required"/>
</complexType>
```

Below is an example instance of the *GridCell* element:

```
<eosps:GridCell gml:id="C01"
  xmlns:eosps="http://www.opengis.net/eosps/2.0"
  xmlns:gml="http://www.opengis.net/gml/3.2">
  <eosps:footprint>
    <gml:Polygon gml:id="FC01" srsName="http://www.opengis.net/def/crs/EPSG/0/4326">
      <gml:exterior>
        <gml:LinearRing>
          <gml:posList>45.1 126.3 46.4 127.5 45.9 128.6 44.8 127.3 45.12 126.3</gml:posList>
        </gml:LinearRing>
      </gml:exterior>
    </gml:Polygon>
  </eosps:footprint>
  <eosps:status>ACQUIRED</eosps:status>
  <eosps:successRate>81</eosps:successRate>
  <eosps:estimatedSuccessDate>2010-10-15T10:10:02Z</eosps:estimatedSuccessDate>
  <eosps:lastAttemptDate>2010-06-01T10:20:32Z</eosps:lastAttemptDate>
  <eosps:nextAttemptDate>2010-06-05T10:22:45Z</eosps:nextAttemptDate>
  <eosps:remainingAttempts>25</eosps:remainingAttempts>
</eosps:GridCell>
```

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/core/gridcell-footprint-srs
Req 25. The value of the *srsName* attribute of the footprint polygon shall be "http://www.opengis.net/def/crs/EPSG/0/4326".

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/core/gridcell-footprint-syntax
Req 26. The footprint polygon shall contain only an exterior ring expressed as a *LinearRing* element which coordinates shall be encapsulated in a *posList* element.

7.1.6.2 Segment Class

7.1.6.2.1 Description

This class is used to provide parameters of one of the segments covering a part of the region of interest. It can be used to provide cell feasibility assessment as well as to provide status of cells coverage.

These segment parameters shall be consistent with the corresponding tasking request. In particular the segment's ground footprint shall intersect the region of interest, the acquisition time shall be within the time period requested and the platform, instrument and sensor information as well as the acquisition parameters shall match the ones requested.

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/core/segment-coherent
Req 27. The attribute values of a *Segment* instance shall be consistent with what was requested in the original feasibility or tasking request.

7.1.6.2.2 Data model

The UML diagram below shows the *Segment* class that is composed of a list of simple parameters giving the characteristics of each segment.

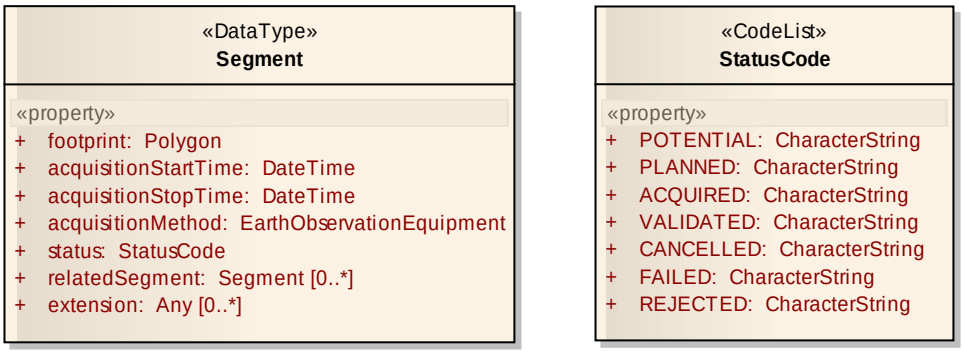

Figure 17 – UML diagram of the *Segment* class

Name	Description	Type (unit)
footprint	Ground footprint of the imagery segment.	Polygon *(OGC 07-036)*
acquisitionStartTime	Date and time at the beginning of the acquisition.	DateTime
acquisitionStopTime	Date and time at the end of the acquisition.	DateTime
acquisitionMethod	Description of the acquisition method including name of instrument and platform, sensor mode, acquisition angles, orbit information, etc.	EarthObservation Equipment *(OGC 10-157)*

status	Status code describing the state of this segment. Descriptions of status codes are given in the table below. The value shall be 'POTENTIAL' in the case of a feasibility study.	StatusCode CodeList
relatedSegment	Reference to a coupled segment such as the other member of a stereo or interferometric pair.	Segment *(by reference)*
extension	Extension slot for mission specific information or future extensions to this standard.	Any

The table below describes the meaning of status codes used in the *Segment* class:

Name	**Description**
POTENTIAL	Segments are marked as potential when they are estimated in the context of a feasibility study.
PLANNED	The segment is planned for acquisition at the specified date. This usually means that the segment has been entered in the satellite work plan but has not yet been executed.
ACQUIRED	The segment was acquired by the on-board instrument and successfully downlinked to a ground station.
VALIDATED	The acquired segment was validated since it satisfied the validation criteria expressed in the request (e.g. cloud cover was less than the maximum acceptable amount).
CANCELLED	The segment acquisition was cancelled by the requester.
REJECTED	An originally planned or potential segment acquisition was purposely cancelled by the data provider because of a last minute conflict.
FAILED	The acquisition of a segment failed for an internal reason such as a last minute conflict or a low image quality resulting from bad downlink transmission.

The *Polygon* class is defined in [OGC 07-036].

The *EarthObservationEquipment* class is defined in [OGC 10-157].

7.1.6.2.3 XML encoding

The following schema snippet normalizes the structure of the *Segment* XML element:

```
<element name="Segment" type="eosps:SegmentType" substitutionGroup="gml:AbstractObject"/>

<complexType name="SegmentType">
  <sequence>
    <element name="footprint" type="gml:SurfacePropertyType"/>
    <element name="acquisitionStartTime" type="dateTime"/>
    <element name="acquisitionStopTime" type="dateTime"/>
    <element name="acquisitionMethod" type="EarthObservationEquipmentPropertyType"/>
    <element name="status" type="eosps:StatusCodeType"/>
    <element name="relatedSegment" type="gml:ReferenceType" minOccurs="0" maxOccurs="unbounded"/>
    <element name="extension" type="anyType" minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
  <attribute ref="gml:id" use="required"/>
</complexType>
```

Below is an example instance of the *Segment* element:

```xml
<eosps:Segment gml:id="S01"
  xmlns:eosps="http://www.opengis.net/eosps/2.0"
  xmlns="http://earth.esa.int/eop"
  xmlns:gml="http://www.opengis.net/gml/3.2">
  <eosps:footprint>
    <gml:Polygon gml:id="FS01" srsName="http://www.opengis.net/def/crs/EPSG/0/4326">
      <gml:exterior>
        <gml:LinearRing>
          <gml:posList></gml:posList>
        </gml:LinearRing>
      </gml:exterior>
    </gml:Polygon>
  </eosps:footprint>
  <eosps:acquisitionStartTime>2010-06-01T10:23:32Z</eosps:acquisitionStartTime>
  <eosps:acquisitionStopTime>2010-06-01T10:23:40Z</eosps:acquisitionStopTime>
  <eosps:acquisitionMethod>
    <EarthObservationEquipment gml:id="E01">
      <platform>
        <Platform>
          <shortName>SPOT-5</shortName>
        </Platform>
      </platform>
      <instrument>
        <Instrument>
          <shortName>HRG1</shortName>
        </Instrument>
      </instrument>
      <sensor>
        <Sensor>
          <operationalMode>THX</operationalMode>
          <resolution uom="m">2.5</resolution>
        </Sensor>
      </sensor>
      <acquisitionParameters>
        <Acquisition>
          <orbitNumber>12</orbitNumber>
          <orbitDirection>DESCENDING</orbitDirection>
          <wrsLongitudeGrid>86</wrsLongitudeGrid>
          <wrsLatitudeGrid>123</wrsLatitudeGrid>
          <incidenceAngle uom="deg">11.5</incidenceAngle>
          <pitch uom="deg">0.0</pitch>
          <roll uom="deg">-10.0</roll>
        </Acquisition>
      </acquisitionParameters>
    </EarthObservationEquipment>
  </eosps:acquisitionMethod>
  <eosps:status>ACQUIRED</eosps:status>
</eosps:Segment>
```

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/core/segment-footprint-syntax
Req 28. The polygon element used to describe the segment's footprint shall comply with Req 25 and Req 26.

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/core/segment-units
Req 29. The unit of measure of all angular quantities used within the *EarthObservationEquipment* element shall be decimal degree ('deg' in UCUM syntax) and the unit of distance shall be meter ('m' in UCUM syntax).

7.1.7 Expected behavior of the GetStatus operation

The sole status report returned in response to a *GetStatus* operation with no '*since*' parameter shall contain all segments acquired since the task was submitted or confirmed. This is always the case when the implementation does not support the "State Logger" conformance class of [OGC 09-000].

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/core/get-status-latest
Req 30. The response to a *GetStatus* request with no '*since*' parameter shall consist of a single status report containing all segments acquired since the task was submitted.

The *GetStatus* operation defined in [OGC 09-000] allows for a '*since*' parameter that enables incremental retrieval of status information if the "State Logger" conformance class is supported. This parameter has to be accounted for correctly by a server implementation of the EO SPS.

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/core/get-status-since
Req 31. The response to a *GetStatus* request with a '*since*' parameter shall contain a status report with only segments and grid cell objects whose status has changed after the date given in the *since* parameter.

7.2 Requirements Class: Additional extensions for optical missions

Requirements Class	
http://www.opengis.net/spec/EOSPS/2.0/req/opt	
Target Type	Server Implementation
Dependency	http://www.opengis.net/spec/EOSPS/2.0/req/core

7.2.1 Introduction

This section defines additional requirements that shall be fulfilled by all EO SPS server implementations supporting tasking of space-borne EO optical sensors. Most requirements are also applicable to client implementations supporting tasking of optical sensors even though no formal conformance testing is defined for clients.

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/opt/dependency-core
Req 32. An EO SPS implementation passing the "Additional extensions for optical missions" conformance class shall first pass the core conformance class.

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/opt/tasking-params-used
Req 33. The EO SPS implementation shall use one of *AcquisitionParametersOPT* or *ValidationParametersOPT* class in its *DescribeTaskingResponse*.

7.2.2 Additional tasking parameters specific to optical missions

7.2.2.1 AcquisitionParametersOPT Class

7.2.2.1.1 Description

This group of parameters is used to specify the acquisition options of an optical instrument in terms of desired resolution, spectral mode, etc.

7.2.2.1.2 Data model

The *AcquisitionParametersOPT* class is derived from the abstract class *AcquisitionParameters* defined in the core of this standard and defines a list of simple attributes.

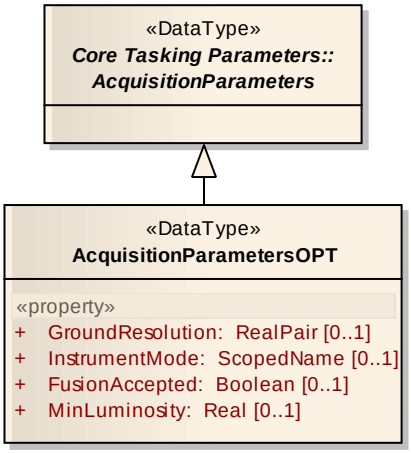

Figure 18 – UML diagram of the *AcquisitionParametersOPT* class

Name	Description	Type (unit)
GroundResolution	The ground resolution is the distance between two contiguous pixels of remote sensed imagery when projected on the earth. This parameter allows the user to select a range of acceptable ground resolutions.	Real Pair (meter)
InstrumentMode	A categorical value used to specify a particular instrument configuration. All possible modes should be defined in a code space, usually specific to the type of instrument.	ScopedName
FusionAccepted	Specifies if imagery obtained by fusing several images acquired separately and co-registered is acceptable.	Boolean
MinLuminosity	Minimum luminosity acceptable for the acquired imagery. This constrains the solar incidence angle of the acquired imagery.	Real (%)

7.2.2.1.3 SWE Common encoding

The following table and XML snippet describe how this tasking parameters group shall be encoded using the SWE Common Data Model:

Field Name	Definition	Component
AcquisitionParameters	http://www.opengis.net/def/property/OGC-EO/0/opt/AcquisitionParametersOPT	DataRecord
GroundResolution	http://www.opengis.net/def/property/OGC-EO/0/GroundResolution	QuantityRange
FusionAccepted	http://www.opengis.net/def/property/OGC-EO/0/FusionAccepted	Boolean
InstrumentMode	http://www.opengis.net/def/property/OGC-EO/0/InstrumentMode A default code space is defined by this specification "http://www.opengis.net/def/property/OGC-EO/0/opt/SpectralModes/" and shall be used when applicable.	Category
MinLuminosity	http://www.opengis.net/def/property/OGC-EO/0/opt/MinLuminosity	Quantity

```
<swe:field name="AcquisitionParameters">
  <swe:DataRecord definition="http://www.opengis.net/def/property/OGC-EO/0/opt/AcquisitionParametersOPT">
    <swe:field name="GroundResolution">
      <swe:QuantityRange definition="http://www.opengis.net/def/property/OGC-EO/0/GroundResolution">
```

```
      <gml:name>Ground Resolution</gml:name>
      <swe:uom code="m"/>
      <swe:constraint>
        <swe:AllowedValues>
          <swe:interval>2.5 20</swe:interval>
        </swe:AllowedValues>
      </swe:constraint>
      <swe:value>2.5 20</swe:value>
    </swe:QuantityRange>
  </swe:field>
  <swe:field name="InstrumentMode">
    <swe:Category definition="http://www.opengis.net/def/property/OGC-EO/0/InstrumentMode">
      <gml:name>Instrument Mode</gml:name>
      <swe:codeSpace xlink:href="http://www.opengis.net/def/property/OGC-EO/0/opt/SpectralModes/"/>
      <swe:constraint>
        <swe:AllowedTokens>
          <swe:value>PANCHROMATIC</swe:value>
          <swe:value>MULTISPECTRAL</swe:value>
        </swe:AllowedTokens>
      </swe:constraint>
      <swe:value>MULTISPECTRAL</swe:value>
    </swe:Category>
  </swe:field>
  <swe:field name="FusionAccepted">
    <swe:Boolean definition="http://www.opengis.net/def/property/OGC-EO/0/FusionAccepted">
      <gml:name>Fusion Accepted</gml:name>
      <swe:value>true</swe:value>
    </swe:Boolean>
  </swe:field>
  <swe:field name="MinLuminosity">
    <swe:Quantity definition="http://www.opengis.net/def/property/OGC-EO/0/opt/MinLuminosity">
      <gml:name>Minimum Luminosity</gml:name>
      <swe:uom code="%"/>
      <swe:constraint>
        <swe:AllowedValues>
          <swe:interval>0 100</swe:interval>
        </swe:AllowedValues>
      </swe:constraint>
      <swe:value>20</swe:value>
    </swe:Quantity>
  </swe:field>
 </swe:DataRecord>
</swe:field>
```

A different instrument mode code space than the one defined in this standard can be used when necessary. In this case, the code space shall be URL accessible and shall always contain the 'MULTISPECTRAL' and 'PANCHROMATIC' options.

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/opt/tasking-params-codespace-gml
Req 34. A custom optical instrument code space shall be URL accessible and encoded in GML 3.2 format. The code space document shall pass the corresponding conformance test class of [OGC 07-036].

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/opt/tasking-params-codespace-default
Req 35. If a custom instrument mode code space is defined, it shall contain the two default options 'MULTISPECTRAL' and 'PANCHROMATIC'.

The following XML snippet illustrates how a custom code space can be used to allow an additional mission specific instrument mode for infrared (SWIR) acquisitions:

```
<swe:field name="InstrumentMode">
  <swe:Category definition="http://www.opengis.net/def/property/OGC-EO/0/InstrumentMode">
    <gml:name>Instrument Mode</gml:name>
    <swe:codeSpace xlink:href="http://www.spotimage.com/registry/InstrumentModes/"/>
    <swe:constraint>
      <swe:AllowedTokens>
        <swe:value>PANCHROMATIC</swe:value>
        <swe:value>MULTISPECTRAL</swe:value>
        <swe:value>XS+SWIR</swe:value>
      </swe:AllowedTokens>
    </swe:constraint>
    <swe:value>MULTISPECTRAL</swe:value>
  </swe:Category>
</swe:field>
```

When connecting to the code space URL, a GML dictionary document containing the definitions of each item in the codespace shall be returned.

7.2.2.2 ValidationParametersOPT Class

7.2.2.2.1 Description

This group of parameters is used to specify the automatic validation options for imagery acquired by an optical instrument. It describes the user needs in terms of image quality. The EO-SPS server shall report an acquired segment as "VALIDATED" (see *ProgrammingStatus* class in section §7.1.5.2) when it satisfies these criteria unless a manual validation mode has been requested (see the *Validate* operation in section §0).

7.2.2.2.2 Data model

The *ValidationParametersOPT* class is derived from the abstract *ValidationParameters* class of the core package and adds specific validation parameters for optical sensors.

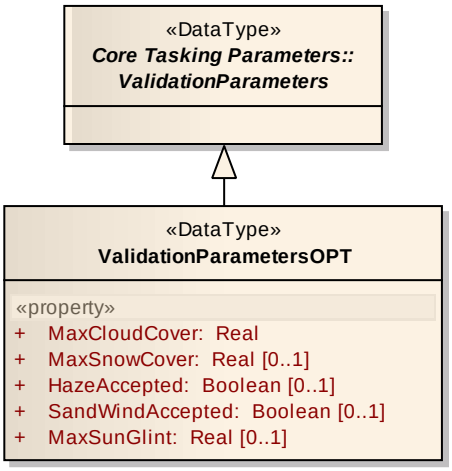

Figure 19 – UML diagram of the *ValidationParametersOPT* class

Name	Description	Type (unit)
MaxCloudCover	Maximum acceptable cloud cover. The cloud cover is the portion of the image area covered by clouds.	Real (%)
MaxSnowCover	Maximum acceptable snow coverage. The snow cover is the portion of the image area covered by snow.	Real (%)
HazeAccepted	Specifies if haze is acceptable.	Boolean
SandWindAccepted	Specifies if sand wind is acceptable.	Boolean
MaxSunGlint	Maximum acceptable portion of the image covered by a sun glint effect.	Real (%)

7.2.2.2.3 SWE Common encoding

The following table and XML snippet describe how this tasking parameters group shall be encoded using the SWE Common Data Model:

Field Name	Definition	Component
ValidationParameters	http://www.opengis.net/def/property/OGC-EO/0/opt/ValidationParametersOPT	DataRecord
MaxCloudCover	http://www.opengis.net/def/property/OGC-EO/0/opt/MaxCloudCover	Quantity
MaxSnowCover	http://www.opengis.net/def/property/OGC-EO/0/opt/MaxSnowCover	Quantity
HazeAccepted	http://www.opengis.net/def/property/OGC-EO/0/opt/HazeAccepted	Boolean
SandWindAccepted	http://www.opengis.net/def/property/OGC-EO/0/opt/SandWindAccepted	Boolean
MaxSunGlint	http://www.opengis.net/def/property/OGC-EO/0/opt/MaxSunGlint	Quantity

```
<swe:field name="ValidationParameters">
  <swe:DataRecord definition="http://www.opengis.net/def/property/OGC-EO/0/opt/ValidationParametersOPT">
    <swe:field name="MaxCloudCover">
      <swe:Quantity definition="http://www.opengis.net/def/property/OGC-EO/0/opt/MaxCloudCover">
        <gml:name>Max Cloud Cover</gml:name>
        <swe:uom code="%"/>
        <swe:constraint>
```



```
        <swe:AllowedValues>
          <swe:interval>0 100</swe:interval>
        </swe:AllowedValues>
      </swe:constraint>
    </swe:Quantity>
  </swe:field>
  <swe:field name="MaxSnowCover" optional="true">
    <swe:Quantity definition="http://www.opengis.net/def/property/OGC-EO/0/opt/MaxSnowCover">
      <gml:name>Max Snow Cover</gml:name>
      <swe:uom code="%"/>
      <swe:constraint>
        <swe:AllowedValues>
          <swe:interval>0 100</swe:interval>
        </swe:AllowedValues>
      </swe:constraint>
    </swe:Quantity>
  </swe:field>
  <swe:field name="HazeAccepted" optional="true">
    <swe:Boolean definition="http://www.opengis.net/def/property/OGC-EO/0/opt/HazeAccepted">
      <gml:name>Haze Accepted</gml:name>
    </swe:Boolean>
  </swe:field>
  <swe:field name="SandWindAccepted" optional="true">
    <swe:Boolean definition="http://www.opengis.net/def/property/OGC-EO/0/opt/SandWindAccepted">
      <gml:name>Sand Wind Accepted</gml:name>
    </swe:Boolean>
  </swe:field>
  <swe:field name="MaxSunGlint" optional="true">
    <swe:Quantity definition="http://www.opengis.net/def/property/OGC-EO/0/opt/MaxSunGlint">
      <gml:name>Max Sun Glint</gml:name>
      <swe:uom code="%"/>
      <swe:constraint>
        <swe:AllowedValues>
          <swe:interval>0 100</swe:interval>
        </swe:AllowedValues>
      </swe:constraint>
    </swe:Quantity>
  </swe:field>
 </swe:DataRecord>
</swe:field>
```

7.2.3 Tasking responses for optical missions

The *Segment* objects listed in feasibility and task status reports and corresponding to imagery segments acquired by optical instruments can make use of the *Acquisition* class defined in the OPT package of the EO Metadata standard [OGC 10-157] instead of the *Acquisition* class defined in the core EOP package.

This class defines additional acquisition metadata specific to optical instruments such as sun illumination angles, etc…

7.3 Requirements Class: Additional extensions for SAR missions

Requirements Class	
http://www.opengis.net/spec/EOSPS/2.0/req/sar	
Target Type	Server Implementation
Dependency	http://www.opengis.net/spec/EOSPS/2.0/req/core

7.3.1 Introduction

This section defines additional requirements that shall be fulfilled by all EO SPS server implementations supporting tasking of space-borne EO SAR sensors. Most requirements are also applicable to client implementations supporting tasking of SAR sensors even though no formal conformance testing is defined for clients.

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/sar/dependency-core
Req 36. An EO SPS implementation passing the "Additional extensions for SAR missions" conformance class shall first pass the core conformance class.

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/sar/tasking-params-used
Req 37. The EO SPS implementation shall use one of *AcquisitionParametersSAR* or *ValidationParametersSAR* class in its *DescribeTaskingResponse*.

7.3.2 Additional tasking parameters specific to radar (SAR) missions

7.3.2.1 AcquisitionParametersSAR Class

7.3.2.1.1 Description

This group of parameters is used to specify the acquisition options of a SAR instrument in terms of desired resolution, acquisition and polarization modes, etc.

7.3.2.1.2 Data model

The *AcquisitionParametersSAR* class is derived from the abstract class *AcquisitionParameters* defined in the core of this standard.

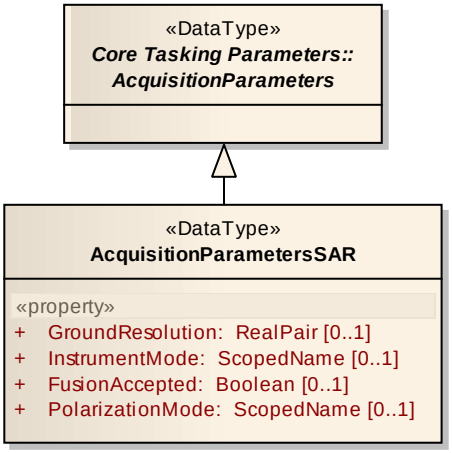

Figure 20 – UML diagram of the *AcquisitionParametersSAR* class

Name	Description	Type (unit)
GroundResolution	The ground resolution is the distance between two contiguous pixels of remote sensed imagery when projected on the earth. This parameter allows the user to select a range of acceptable ground resolutions.	Real Pair (meter)
InstrumentMode	A categorical value used to specify a particular instrument configuration. All possible modes should be defined by a code space, usually specific to the instrument in the case of SAR.	ScopedName
FusionAccepted	Specifies if imagery obtained by fusing several images acquired separately and co-registered is acceptable.	Boolean
PolarizationMode	Receive/Transmit polarization mode of a SAR instrument.	ScopedName

7.3.2.1.3 SWE Common encoding

The following table and XML snippet describe how this tasking parameters group shall be encoded using the SWE Common Data Model:

Field Name	Definition	Component
AcquisitionParameters	http://www.opengis.net/def/property/OGC-EO/0/sar/AcquisitionParametersSAR	DataRecord
GroundResolution	http://www.opengis.net/def/property/OGC-EO/0/GroundResolution	QuantityRange
InstrumentMode	http://www.opengis.net/def/property/OGC-EO/0/InstrumentMode No default code space is defined by this specification for SAR instruments since there does not seem to be a standard for naming SAR acquisition modes. A URL linking to the definitions of each listed mode shall thus be provided by each SAR implementation of EO-SPS.	Category
FusionAccepted	http://www.opengis.net/def/property/OGC-EO/0/FusionAccepted	Boolean
PolarizationMode	http://www.opengis.net/def/property/OGC-EO/0/sar/PolarizationMode A default code space is defined by this specification "http://www.opengis.net/def/property/OGC-EO/0/sar/PolarizationModes/" and should be used when applicable	Category

```
<swe:field name="AcquisitionParameters">
  <swe:DataRecord definition="http://www.opengis.net/def/property/OGC-EO/0/sar/AcquisitionParametersSAR">
    <swe:field name="GroundResolution">
      <swe:QuantityRange definition="http://www.opengis.net/def/property/OGC-EO/0/GroundResolution">
        <gml:name>Ground Resolution</gml:name>
        <swe:uom code="m"/>
        <swe:constraint>
          <swe:AllowedValues>
            <swe:interval>1.0 50</swe:interval>
          </swe:AllowedValues>
        </swe:constraint>
        <swe:value>20 50</swe:value>
      </swe:QuantityRange>
    </swe:field>
    <swe:field name="InstrumentMode">
      <swe:Category definition="http://www.opengis.net/def/property/OGC-EO/0/InstrumentMode">
        <gml:name>Instrument Mode</gml:name>
        <swe:codeSpace xlink:href="http://www.dlr.de/registry/TXModes/"/>
        <swe:constraint>
          <swe:AllowedTokens>
            <swe:value>ScanSAR</swe:value>
            <swe:value>StripMap</swe:value>
            <swe:value>SpotLight</swe:value>
          </swe:AllowedTokens>
        </swe:constraint>
        <swe:value>ScanSar</swe:value>
      </swe:Category>
    </swe:field>
    <swe:field name="FusionAccepted">
      <swe:Boolean definition="http://www.opengis.net/def/property/OGC-EO/0/FusionAccepted">
        <gml:name>Fusion Accepted</gml:name>
        <swe:value>true</swe:value>
      </swe:Boolean>
    </swe:field>
    <swe:field name="PolarizationMode">
      <swe:Category definition="http://www.opengis.net/def/property/OGC-EO/0/sar/PolarizationMode">
        <gml:name>Polarization Mode</gml:name>
        <swe:codeSpace xlink:href="http://www.opengis.net/def/property/OGC-EO/0/sar/PolarizationModes/"/>
        <swe:constraint>
          <swe:AllowedTokens>
            <swe:value>HH</swe:value>
            <swe:value>VV</swe:value>
            <swe:value>HH-HV</swe:value>
            <swe:value>VV-VH</swe:value>
          </swe:AllowedTokens>
        </swe:constraint>
        <swe:value>HH</swe:value>
      </swe:Category>
    </swe:field>
  </swe:DataRecord>
</swe:field>
```

The code space defining all instrument modes (identified by http://www.dlr.de/registry/TXModes/ in the example above) shall be provided in a URL accessible form and encoded in the GML format. An example of instrument mode code space is given in Annex C.

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/sar/tasking-params-codepace
Req 38. A code space identifying and defining the supported SAR instrument modes shall be provided.

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/sar/tasking-params-codespace-gml
Req 39. A custom SAR code space shall be URL accessible and encoded in GML 3.2 format. The code space document shall pass the corresponding conformance test class of [OGC 07-036].

7.3.2.2 ValidationParametersSAR Class

7.3.2.2.1 Description

This group of parameters is used to specify the automatic validation options for imagery acquired by a SAR instrument. It describes the user needs in terms of image quality. The EO-SPS server shall report an acquired segment as "VALIDATED" (see *ProgrammingStatus* class in section §7.1.5.2) when it satisfies these criteria unless a manual validation mode has been requested (see the Validate operation in section §0).

7.3.2.2.2 Data model

The *ValidationParametersSAR* class is derived from the abstract *ValidationParameters* class of the core package and adds specific validation parameters for SAR sensors.

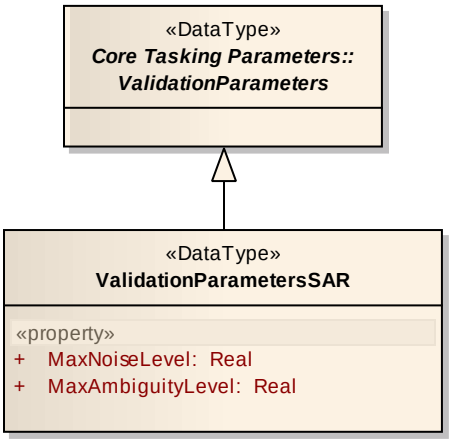

Figure 21 – UML diagram of the *ValidationParametersSAR* class

Name	Description	Type (unit)
MaxNoiseLevel	Maximum acceptable signal to noise ratio for each acquired imagery segment.	Real (decibel)
MaxAmbiguityLevel	Maximum acceptable ambiguity level for each acquired imagery segment.	Real (decibel)

7.3.2.2.3 SWE Common encoding

The following table and XML snippet describe how this tasking parameters group shall be encoded using the SWE Common Data Model:

Field Name	Definition	Component
ValidationParameters	http://www.opengis.net/def/property/OGC-EO/0/sar/ValidationParametersSAR	DataRecord
MaxNoiseLevel	http://www.opengis.net/def/property/OGC-EO/0/sar/MaxNoiseLevel	Quantity
MaxAmbiguityLevel	http://www.opengis.net/def/property/OGC-EO/0/sar/MaxAmbiguityLevel	Quantity

```
<swe:field name="ValidationParameters">
  <swe:DataRecord definition="http://www.opengis.net/def/property/OGC-EO/0/sar/ValidationParametersSAR">
    <swe:field name="MaxNoiseLevel">
      <swe:Quantity definition="http://www.opengis.net/def/property/OGC-EO/0/sar/MaxNoiseLevel">
        <gml:name>Max Noise Level</gml:name>
        <swe:uom code="dB"/>
      </swe:Quantity>
    </swe:field>
    <swe:field name="MaxAmbiguityLevel">
      <swe:Quantity definition="http://www.opengis.net/def/property/OGC-EO/0/sar/MaxAmbiguityLevel">
        <gml:name>Max Ambiguity Level</gml:name>
        <swe:uom code="dB"/>
      </swe:Quantity>
    </swe:field>
  </swe:DataRecord>
</swe:field>
```

7.3.3 Tasking responses for SAR missions

The *Segment* objects listed in feasibility and task status reports and corresponding to imagery segments acquired by SAR instruments can make use of the *Acquisition* class defined in the SAR package of the EO Metadata standard [OGC 10-157] instead of the *Acquisition* class defined in the core EOP package.

This class defines additional acquisition metadata specific to SAR instruments such as polarization mode and channels, antenna look direction, etc…

8 Additional operations and SOAP binding

8.1 Requirements Class: GetSensorAvailability Operation

Requirements Class	
http://www.opengis.net/spec/EOSPS/2.0/req/gsa	
Target Type	Server Implementation
Dependency	http://www.opengis.net/spec/EOSPS/2.0/req/core

8.1.1 Introduction

This section defines an additional operation that can be implemented by EO SPS servers and clients. The standardization target of this class is a server implementation supporting retrieval of future sensor availability but most requirements are also applicable to client implementations seeking to support this feature.

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/gsa/capabilities
Req 40. The *GetSensorAvailability* operation shall be listed in the capabilities document of the service.

An EO system may not be available over a period of time for different reasons such as workload, maintenance, etc. The *GetSensorAvailability* operation allows the client to obtain a preview of the periods of availability of a sensor before a feasibility study is requested.

The granularity of the provided information is up to the data provider who can choose to describe its exact workload or simply list approximate periods of availability. For instance, a provider may wish to list a period of one week as soon as the sensor is available for tasking at least during 50% of the period. This allows a provider to choose the most appropriate granularity of information to help the users while maintaining its exact workload secret.

8.1.2 Data model

The following UML diagram shows the exact model of the *GetSensorAvailability* operation:

Figure 22 – UML diagram of the *GetSensorAvailability* operation

8.1.2.1 Operation request – GetSensorAvailability

Sending an instance of the *GetSensorAvailability* data type to the service performs an SPS EO *GetSensorAvailability* operation request.

The *GetSensorAvailability* data type is derived from the *ExtensibleRequest* data type specified in the SWE Service Model standard [OGC 09-001] and therefore inherits all the properties contained in that data type. *GetSensorAvailability* does not restrict the content model of *ExtensibleRequest*. It shall contain the properties defined for *ExtensibleRequest*. In addition, it shall include the properties listed in the following table.

Name	Definition	Data Type	Multiplicity
sensor	Identifier of a sensor/procedure as advertised in the capabilities.	ScopedName	One (mandatory)
requestPeriod	Period during which the client asks for sensor availability.	TM_Period	One[1] (mandatory)

1. The number of periods that can be sent to the server by the client is voluntary restricted to one because the *GetSensorAvailability* operation is meant to give the client an overall view of the sensor availability over a certain period of time. If a more precise information is needed the *GetFeasibility* operation should be used.

8.1.2.2 Operation response – GetSensorAvailabilityResponse

The *GetSensorAvailabilityResponse* data type represents the response to an SPS EO *GetSensorAvailability* operation request.

The *GetSensorAvailabilityResponse* data type is derived from the *ExtensibleResponse* data type defined in [OGC 09-001] and therefore inherits all the properties contained in

that data type. *GetSensorAvailabilityResponse* does not restrict the content model of *ExtensibleResponse*, but adds the additional properties that contain the availability time periods. It shall therefore contain the properties defined for the *ExtensibleResponse* data type plus the types defined in the following table.

Name	Definition	Data Type	Multiplicity
responsePeriod	Period for which availability has been assessed by the server. It can be different from the requested period as explained below.	TM_Period	One (mandatory)
availabilityPeriod	Period(s) of availability of the sensor. When no periods are listed, the sensor is not available at all during the response period.	TM_Period	Zero or more

Some missions can provide many details about sensor availability while some others cannot, so that there are no special requirements on the granularity and the accuracy of the availability periods returned. The service is free to choose the level of details of the response. For example, one global period or several short periods of availability can be returned and either precise time range with hours or date only can be used.

For the same reason, the server may not trim availability periods to the exact period of time given by the client in the request since that would allow for precise discovery of the system's workload. Typically, when the requested period is small, a larger period than the one requested can be included in the response in order to prevent clients from discovering more precise information by issuing multiple well-chosen requests.

The *responsePeriod* also indicates the period for which availability was really assessed. For instance, when the requested period is large (i.e. more than one year in the future), the ground segment may not be available to provide information on the system availability for the whole period. Instead, the server can describe its availability only up to its maximum planning horizon and indicate this with the *responsePeriod* attribute in the response.

8.1.3 XML Encoding

The following XML schema snippet normalizes the XML encoding of the *GetSensorAvailability class*.

```
<element name="GetSensorAvailability" type="eosps:GetSensorAvailabilityType"/>

<complexType name="GetSensorAvailabilityType">
  <complexContent>
    <extension base="swes:ExtensibleRequestType">
      <sequence>
        <element name="sensor" type="anyURI"/>
        <element name="requestPeriod">
          <complexType>
            <sequence>
              <element ref="gml:TimePeriod"/>
            </sequence>
```

```
        </complexType>
      </element>
    </sequence>
  </extension>
 </complexContent>
</complexType>
```

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/gsa/request-valid
Req 41. The XML message encoding a *GetSensorAvailability* request shall be valid with respect to the getSensorAvailability.xsd XML schema. The server shall correctly reject an invalid request by returning the proper correctly formatted exception as defined in [OGC 06-121] and [OGC 09-001].

The following XML schema snippet normalizes the XML encoding of the *GetSensorAvailabilityResponse class*.

```
<element name="GetSensorAvailabilityResponse" type="eosps:GetSensorAvailabilityResponseType"/>

<complexType name="GetSensorAvailabilityResponseType">
 <complexContent>
  <extension base="swes:ExtensibleResponseType">
   <sequence>
    <element name="responsePeriod">
     <complexType>
      <sequence>
       <element ref="gml:TimePeriod"/>
      </sequence>
     </complexType>
    </element>
    <element name="availabilityPeriod" minOccurs="0" maxOccurs="unbounded">
     <complexType>
      <sequence>
       <element ref="gml:TimePeriod"/>
      </sequence>
     </complexType>
    </element>
   </sequence>
  </extension>
 </complexContent>
</complexType>
```

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/gsa/response-valid
Req 42. The XML message encoding a *GetSensorAvailabilityResponse* shall be valid with respect to the getSensorAvailability.xsd XML schema.

Below is an example *GetSensorAvailability* request along with the corresponding response:

```
<GetSensorAvailability service="EOSPS" version="2.0"
 xmlns="http://www.opengis.net/eosps/2.0"
 xmlns:gml="http://www.opengis.net/gml/3.2">
```



```
  <sensor>http://ws.spotimage.com/sps/sensors/SPOT-Constellation</sensor>
  <requestPeriod>
    <gml:TimePeriod gml:id="PERIOD">
      <gml:beginPosition>2010-06-01</gml:beginPosition>
      <gml:endPosition>2010-08-31</gml:endPosition>
    </gml:TimePeriod>
  </requestPeriod>
</GetSensorAvailability>

<GetSensorAvailabilityResponse
  xmlns="http://www.opengis.net/eosps/2.0"
  xmlns:gml="http://www.opengis.net/gml/3.2">
  <responsePeriod>
    <gml:TimePeriod gml:id="PERIOD">
      <gml:beginPosition>2010-06-01</gml:beginPosition>
      <gml:endPosition>2010-08-31</gml:endPosition>
    </gml:TimePeriod>
  </responsePeriod>
  <availabilityPeriod>
    <gml:TimePeriod gml:id="A1">
      <gml:beginPosition>2010-06-21</gml:beginPosition>
      <gml:endPosition>2010-07-04</gml:endPosition>
    </gml:TimePeriod>
  </availabilityPeriod>
  <availabilityPeriod>
    <gml:TimePeriod gml:id="A2">
      <gml:beginPosition>2010-08-21</gml:beginPosition>
      <gml:endPosition>2010-08-28</gml:endPosition>
    </gml:TimePeriod>
  </availabilityPeriod>
</GetSensorAvailabilityResponse>
```

8.1.4 Exceptions

When an SPS EO server encounters an error while performing a *GetSensorAvailability* operation, it shall return an exception report message as specified in the Sensor Planning Service 2.0 standard [OGC 09-000].

The following table presents exception codes that are applicable to the *GetSensorAvailability* operation:

Exception Codes	Applicable
OperationNotSupported	**Yes**
MissingParameterValue	**Yes**
InvalidParameterValue	**Yes**
VersionNegotiationFailed	**No**
InvalidUpdateSequence	**No**
OptionNotSupported	**No**
NoApplicableCode	**Yes**
InvalidRequest	**Yes**

An *InvalidParameterValue* exception shall be returned when the sensor identifier is not recognized by the service.

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/gsa/sensor-id-valid
Req 43. The *sensor* parameter of a *GetSensorAvailability* request shall contain the identifier of a sensor listed in the capabilities. If this is not the case, the server shall return an exception with the code "InvalidParameterValue" as specified in [OGC 06-121].

8.2 Requirements Class: Validate Operation

Requirements Class	
http://www.opengis.net/spec/EOSPS/2.0/req/val	
Target Type	Server Implementation
Dependency	http://www.opengis.net/spec/EOSPS/2.0/req/core

8.2.1 Introduction

This section defines an additional operation that can be implemented by EO SPS servers and clients. The standardization target of this class is a server implementation supporting client-side validation of acquired data segments but most requirements are also applicable to client implementations seeking to support this feature.

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/val/capabilities
Req 44. The *Validate* operation shall be listed in the capabilities document of the service.

Several attempts are sometimes needed for earth observation systems to acquire satisfactory data for an area of the earth surface. For instance, optical acquisitions are subject to cloud cover that can prevent clear imaging of the region of interest. In this case new acquisitions are performed until the region of interest is image properly.

The desired image quality is expressed when sending the programming request to the service so that the data provider can automatically select acquisitions that match the validation criteria. However, some users want to control this validation step manually. The validate operation fulfills this need by allowing the user to stop further acquisitions over a given cell when he is satisfied with the image quality or to request continuation of acquisitions when he is not.

8.2.2 Data model

The following UML diagram shows the exact model of the *Validate* operation:

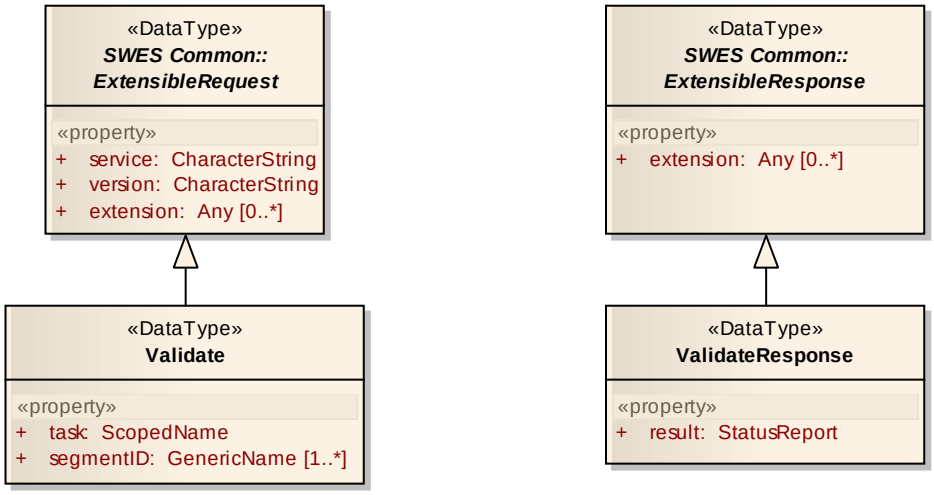

Figure 23 – UML diagram of the *Validate* operation

8.2.2.1 Operation request – Validate

Sending an instance of the Validate data type to the service performs an SPS EO Validate operation request.

The *Validate* data type is derived from the *ExtensibleRequest* data type specified in the SWE Service Model standard [OGC 09-001] and therefore inherits all the properties contained in that data type. *Validate* does not restrict the content model of *ExtensibleRequest*. It shall contain the properties defined for *ExtensibleRequest*. In addition, it shall include the properties listed in the following table.

Name	Definition	Data Type	Multiplicity
task	Identifier of the task that contains the segment to be validated.	ScopedName, value as provided in *StatusReport*.	One (mandatory)
segmentID	Identifier of the acquisition segment to be validated. All segments selected for validation shall have the status ACQUIRED.	GenericName, shall correspond to the ID of Segment objects listed in the *ProgrammingStatus*.	One or more (mandatory)

8.2.2.2 Operation response – ValidateResponse

The *ValidateResponse* data type represents the response to an SPS EO *Validate* operation request. It shall contain a *StatusReport* object similarly to many SPS operations. The status of all segment identified in the request shall be switched to ‘VALIDATED’ unless an error occurred.

The *ValidateResponse* data type is derived from the *ExtensibleResponse* data type defined in [OGC 09-001] and therefore inherits all the properties contained in that data type. *ValidateResponse* does not restrict the content model of *ExtensibleResponse*, but adds the additional status property that contains the reports. It shall therefore contain the properties defined for the *ExtensibleResponse* data type plus the types defined in the following table.

Name	Definition	Data Type	Multiplicity
result	Current status report eventually reflecting validation of the selected segments if the operation is executed synchronously.	StatusReport *(from OGC 09-000)*	One (mandatory)

8.2.3 XML Encoding

The following XML schema snippet normalizes the XML encoding of the *ValidateResponse class*.

```
<element name="Validate" type="eosps:ValidateType" substitutionGroup="swes:ExtensibleRequest"/>

<complexType name="ValidateType">
  <complexContent>
    <extension base="swes:ExtensibleRequestType">
      <sequence>
        <element name="task" type="anyURI"/>
        <element name="segmentID" type="NCName" maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/val/request-valid
Req 45. The XML message encoding a *Validate* request shall be valid with respect to the validate.xsd XML schema. The server shall correctly reject an invalid request by returning the proper correctly formatted exception as defined in [OGC 06-121] and [OGC 09-001].

The following XML schema snippet normalizes the XML encoding of the *ValidateResponse class*.

```
<element name="ValidateResponse" type="eosps:ValidateResponseType"
substitutionGroup="swes:ExtensibleResponse"/>

<complexType name="ValidateResponseType">
  <complexContent>
    <extension base="swes:ExtensibleResponseType">
      <sequence>
        <element name="result" type="sps:StatusReportPropertyType"/>
      </sequence>
    </extension>
```

```
  </complexContent>
</complexType>
```

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/val/response-valid
Req 46. The XML message encoding a *ValidateResponse* shall be valid with respect to the validate.xsd XML schema.

Below is an example of *Validate* request and the corresponding response:

```
<Validate service="EOSPS" version="2.0"
  xmlns="http://www.opengis.net/eosps/2.0">
  <task>http://ws.spotimage.com/sps/tasks/T45EF36C23D4C89AB</task>
  <segmentID>S25</segmentID>
  <segmentID>S26</segmentID>
</Validate>

<ValidateResponse xmlns="http://www.opengis.net/eosps/2.0" xmlns:sps="http://www.opengis.net/sps/2.0">
  <result>
    <sps:StatusReport ... />
  </result>
</ValidateResponse>
```

8.2.4 Exceptions

When an SPS EO server encounters an error while performing a *Validate* operation, it shall return an exception report message as specified in the Sensor Planning Service 2.0 standard [OGC 09-000].

The following table presents exception codes that are applicable to the *Validate* operation.

Exception Codes	Applicable
OperationNotSupported	**Yes**
MissingParameterValue	**Yes**
InvalidParameterValue	**Yes**
VersionNegotiationFailed	**No**
InvalidUpdateSequence	**No**
OptionNotSupported	**No**
NoApplicableCode	**Yes**
InvalidRequest	**Yes**

An *InvalidParameterValue* exception shall be returned when the task identifier is not recognized by the service.

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/val/task-id-valid
Req 47. The *task* parameter of a *Validate* request shall contain the identifier of an existing task. If this is not the case, the server shall return an exception with the code "InvalidParameterValue" as specified in [OGC 06-121].

An *InvalidParameterValue* exception shall be returned when segment identifiers listed in the request do not correspond to the ones listed in the progress report of the task, or when these IDs correspond to segments with a status different than 'ACQUIRED'.

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/val/segment-id-valid
Req 48. The *segmentID* parameter of a *Validate* request shall contain the identifier of an acquired segment listed in the status report of the selected task. If this is not the case, the server shall return an exception with the code "InvalidParameterValue" as specified in [OGC 06-121].

8.2.5 ManualValidation request extension element

8.2.5.1 Description

The *ManualValidation* tag can be inserted within a *Submit* or *Reserve* request to indicate that the client will manually validate newly acquired segments for his task.

When the value of this parameter is 'false' or when it is completely omitted, the provider is responsible for validating the acquired segment when their characteristics match the specified validation criteria. The client thus does not need to act when new segments are received and their status will automatically switch to 'VALIDATED' if the acquired image is acceptable.

When the value of this parameter is 'true', the client is responsible for calling the *Validate* operation (see section §0) to explicitly validate each newly acquired imagery segment. The programming service will usually not attempt to cover the area again until the client calls the operation.

8.2.5.2 XML encoding

The following schema snippet shows the global element declaration that normalizes the encoding of this parameter in XML:

```
<element name="ManualValidation" type="boolean"/>
```

This parameter shall be encapsulated in an extension slot of a *Submit* or *Reserve* request as show below:

```
<sps:Submit service="SPS" version="2.0"
  xmlns:sps="http://www.opengis.net/sps/2.0"
  xmlns:swes="http://www.opengis.net/swes/2.0"
  xmlns:eosps="http://www.opengis.net/eosps/2.0">
  <swes:extension>
    <eosps:ManualValidation>true</eosps:ManualValidation>
  </swes:extension>
    ...
</sps:Submit>
```

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/val/manual-validation-valid
Req 49. The *ManualValidation* XML element shall be valid with respect to the spsRequestExtensions.xsd schema and inserted within an extension element of the Submit or Reserve XML request.

8.3 Requirements Class: SubmitSegmentByID Operation

Requirements Class	
http://www.opengis.net/spec/EOSPS/2.0/req/sid	
Target Type	Server Implementation
Dependency	http://www.opengis.net/spec/EOSPS/2.0/req/core

8.3.1 Introduction

This section defines an additional operation that can be implemented by EO SPS servers and clients. The standardization target of this class is a server implementation supporting the selection of individual segments for tasking but most requirements are also applicable to client implementations seeking to support this feature.

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/sid/capabilities
Req 50. The *SubmitSegmentByID* operation shall be listed in the capabilities document of the service.

The *SubmitSegmentByID* operation can be used by deterministic missions to allow the user to request the acquisition of only some of potential segments identified in a feasibility study report.

8.3.2 Data model

Like for all other SPS operations, the *SubmitSegmentByID* request and response classes are derived from common abstract classes defined in the SWE Service Model standard and inherited from the SPS standard. The following UML diagram shows the exact model of the operation:

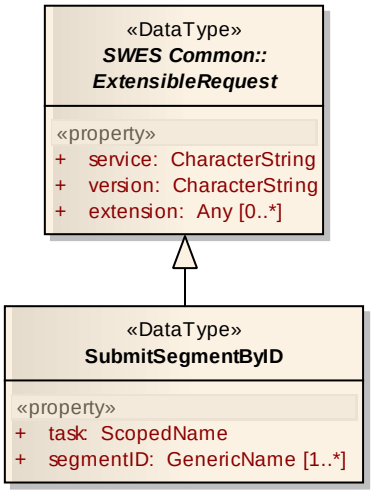

Figure 24 – UML diagram of the *SubmitSegmentByID* operation

8.3.2.1 Operation request – SubmitSegmentByID

Sending an instance of the *SubmitSegmentByID* data type to the service performs an SPS EO *SubmitSegmentByID* operation request.

The *SubmitSegmentByID* data type is derived from the *ExtensibleRequest* data type specified in the SWE Service Model standard [OGC 09-001] and therefore inherits all the properties contained in that data type. *SubmitSegmentByID* does not restrict the content model of *ExtensibleRequest*. It shall contain the properties defined for *ExtensibleRequest*. In addition, it shall include the properties listed in the following table.

Name	Definition	Data Type	Multiplicity
task	Identifier of the task that contains the segment to be acquired.	ScopedName, value as provided in *StatusReport*.	One (mandatory)
segmentID	Identifier of the planned or potential segment to be acquired.	GenericName, shall correspond to the ID of Segment objects listed in a *FeasibilityStudy* object.	One or more (mandatory)

8.3.2.2 Operation response – SubmitResponse

The response to the *SubmitSegmentByID* request shall be a *SubmitResponse* data type as defined in the Sensor Planning Service standard.

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/sid/response-valid
Req 51. The response to the *SubmitSegmentByID* operation shall be an instance of the *SubmitResponse* class defined in [OGC 09-000].

8.3.3 XML Encoding

The following XML schema snippet normalizes the XML encoding of the *SubmitSegmentByID class*.

```
<element name="SubmitSegmentByID" type="eosps:SubmitSegmentByIDType"
substitutionGroup="swes:ExtensibleRequest"/>

<complexType name="SubmitSegmentByIDType">
  <complexContent>
    <extension base="swes:ExtensibleRequestType">
      <sequence>
        <element name="task" type="gml:CodeType"/>
        <element name="segmentID" type="gml:CodeType" maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/sid/request-valid
Req 52. The XML message encoding a *SubmitSegmentByID* request shall be valid with respect to the submitSegmentByID.xsd XML schema. The server shall correctly reject an invalid request by returning the proper correctly formatted exception as defined in [OGC 06-121] and [OGC 09-001].

Below is an example of *SubmitSegmentByID* request and the corresponding response:

```
<SubmitSegmentByID service="EOSPS" version="2.0"
  xmlns="http://www.opengis.net/eosps/2.0">
  <task>http://earth.esa.int/sps/feasibility/F45EF36C23D4C89AB</task>
  <segmentID>S03</segmentID>
  <segmentID>S06</segmentID>
</SubmitSegmentByID>
```

8.3.4 Exceptions

When an SPS EO server encounters an error while performing a *SubmitSegmentByID* operation, it shall return an exception report message as specified in the Sensor Planning Service 2.0 standard [OGC 09-000].

The following table presents exception codes that are applicable to the *SubmitSegmentByID* operation.

Exception Codes	Applicable
OperationNotSupported	**Yes**
MissingParameterValue	**Yes**
InvalidParameterValue	**Yes**
VersionNegotiationFailed	**No**
InvalidUpdateSequence	**No**
OptionNotSupported	**No**
NoApplicableCode	**Yes**
InvalidRequest	**Yes**

An *InvalidParameterValue* exception shall be returned when the task identifier does not correspond to an existing non-expired feasibility study.

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/sid/task-id-valid
Req 53. The *task* parameter of a *SubmitSegmentByID* request shall contain the identifier of an existing and active feasibility study. If this is not the case, the server shall return an exception with the code "InvalidParameterValue" as specified in [OGC 06-121].

An *InvalidParameterValue* exception shall be returned when segment identifiers listed in the request do not correspond to the ones listed in a previously generated feasibility report and with a status different from 'REJECTED'.

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/sid/segment-id-valid
Req 54. The *segmentID* parameter of a *SubmitSegmentByID* request shall contain the identifier of a feasible segment listed in the status report of the selected feasibility study. If this is not the case, the server shall return an exception with the code "InvalidParameterValue" as specified in [OGC 06-121].

8.4 Requirements Class: SOAP binding

Requirements Class	
http://www.opengis.net/spec/EOSPS/2.0/req/soap	
Target Type	Server Implementation
Dependency	http://www.opengis.net/spec/EOSPS/2.0/req/core
Dependency	http://www.opengis.net/doc/IS/SPS/2.0/clause/9

8.4.1 Introduction

This class defines additional requirements so that operations defined in this standard can be accessible via the HTTP/SOAP binding. Implementations seeking conformance to this requirements class shall implement the SOAP binding for all operations defined in [OGC 09-000] as well as the ones defined in this document and supported by the service.

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/soap/all-operations
Req 55. The EO SPS implementation shall allow access to all supported operations via the SOAP binding defined in [OGC 09-000].

All requirements related to the SOAP binding defined in the SPS 2.0 standard [OGC 09-000] are applicable. In particular, exceptions shall be returned in the format specified and the correct SOAP action URIs shall be used with the corresponding message facets without changes.

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/soap/dependency-sps
Req 56. An EO SPS implementation passing the "SOAP binding" conformance class shall first pass the "SOAP" conformance class of [OGC 09-000].

8.4.2 Action URIs

When using the SOAP binding to access the operations defined in this standard, the SOAP actions listed in the following table shall be used:

Message Facet	SOAP Action URI
GetSensorAvailability	http://www.opengis.net/eosps/2.0/GetSensorAvailability
GetSensorAvailabilityResponse	http://www.opengis.net/eosps/2.0/GetSensorAvailabilityResponse
Validate	http://www.opengis.net/eosps/2.0/Validate

ValidateResponse	http://www.opengis.net/eosps/2.0/ValidateResponse
SubmitSegmentByID	http://www.opengis.net/eosps/2.0/SubmitSegmentByID

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/soap/action-uris
Req 57. The proper SOAP action URI shall be used when calling EO SPS operations via the SOAP binding (version 1.1 or 1.2).

8.4.3 Asynchronous call to GetFeasibility with WS-Addressing

When the SOAP binding is implemented, asynchronous behavior of the *GetFeasibility* operation using WS-Addressing is required to be implemented in addition to the synchronous version and in accordance with the *ReplyTo* mechanism defined in the WS-Addressing standard from W3C [W3C WS-A].

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/soap/gf-wsa-support
Req 58. The EO SPS implementation shall support asynchronous access to the *GetFeasibility* operation by using WS-Addressing *ReplyTo* mechanism.

When the *GetFeasibility* operation is called with a WS-Addressing *ReplyTo* element in the SOAP header, the server shall send a single *GetFeasibiltyResponse* to the *ReplyTo* URL and this response shall contain the final feasibility study report.

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/soap/gf-wsa-single-response
Req 59. When the header of the SOAP *GetFeasibility* request contains the ReplyTo element, a single response shall be sent to the *ReplyTo* URL.

9 Extensions to the SPS notification system

9.1 Introduction

The Sensor Planning Service 2.0 standard [OGC 09-000] defines a state machine for a task's life cycle as well as provides a mechanism for subscribing to notifications when certain events (i.e. state changes) occur during this lifecycle. Subscription and reception of notifications is done via the WS-Notification protocol which shall also be implemented by any SPS server wishing to publish notifications to its users. This extension leverages the use of existing SPS events and WS-Notification to publish events related to the operation of EO satellites.

9.2 Requirements Class: Notifications

Requirements Class	
http://www.opengis.net/spec/EOSPS/2.0/req/notif	
Target Type	Server Implementation
Dependency	http://www.opengis.net/spec/EOSPS/2.0/req/core
Dependency	http://www.opengis.net/doc/IS/SPS/2.0/clause/8

WS-Notification defines a flexible mechanism by which a service provider can advertise the types of events a client can subscribe too. This mechanism uses so called "Topics" to describe the list of such events. The Sensor Planning Service 2.0 standard provides a predefined list of such topics that are common to any SPS implementation and that correspond to all possible task state transitions.

All notification messages shall be generated when the corresponding task state transition occurs as specified in [OGC 09-000].

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/notif/dependency-sps
Req 60. An EO SPS implementation passing the "Notifications" conformance test class shall first pass the "Channel Based PubSub" conformance test class defined in the Sensor Planning Service 2.0 Interface Standard [OGC 09-000].

The following events shall be advertised and correctly generated by the EO SPS implementation: *TaskRequestAccepted*, *TaskRequestRejected*, *TaskSubmitted*, *TaskCompleted*, *TaskFailed* and *DataPublished*.

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/notif/events-support-basic
Req 61. The EO SPS implementation shall advertise and generate notifications for events *TaskRequestAccepted, TaskRequestRejected, TaskSubmitted, TaskCompleted, TaskFailed and DataPublished*.

If the implementation supports the *Reserve* and *Confirm* operations, the following events shall also be supported: *TaskReserved*, *ReservationExpired* and *TaskConfirmed*.

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/notif/events-support-reserve
Req 62. An EO SPS implementation supporting task reservation shall advertise and generate notifications for the following events: *TaskReserved, TaskConfirmed and ReservationExpired.*

If the implementation supports the *Cancel* operation, the following event shall also be supported: *TaskCancelled*.

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/notif/events-support-cancel
Req 63. An EO SPS implementation supporting task cancellation shall advertise and generate notifications for the event *TaskCancelled.*

If the implementation supports the *Update* operation, the following event shall also be supported: *TaskUpdated*.

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/notif/events-support-update
Req 64. An EO SPS implementation supporting task updates shall advertise and generate notifications for the event *TaskUpdated.*

Additionally, this standard defines several additional notification topics that are specific to earth observation satellites programming. These topics correspond to events occurring within the '*InExecution*' state. They shall be supported by the EO SPS implementation and are described below:

- **SegmentPlanned**: Notifications in this topic shall be generated when at least one new segment is added to the task status with the state 'PLANNED'.

- **SegmentAcquired**: Notifications in this topic shall be generated when the state of at least one segment changes to 'ACQUIRED' or is added to the task status with the state 'PLANNED'.

- **SegmentValidated**: Notifications in this topic shall be generated when the state of at least one segment changes to 'VALIDATED'.

- **SegmentCancelled**: Notifications in this topic shall be generated when the state of at least one segment that was previously 'PLANNED' changes to 'CANCELLED'.

- **SegmentFailed**: Notifications in this topic shall be generated when the state of at least one segment that was previously 'PLANNED' changes to 'FAILED'

Requirement
http://www.opengis.net/spec/EOSPS/2.0/req/notif/events-support-eo
Req 65. The EO SPS implementation shall advertise and generate notifications for the events *SegmentPlanned*, *SegmentAcquired*, *SegmentValidated*, *SegmentCancelled* and *SegmentFailed*.

Note that these events are not reflected in the change of status of the task itself which remains '*InExecution*', but by the change of status of individual segments.

Note: It is recommended that implementations try to aggregate several events of the same type (i.e. same WS-N topic) happening closely in time and on the same task in a single notification message, as this can help reducing the number of notification messages sent. For example when several segments are acquired almost simultaneously, it is recommended to send a single SegmentAcquired event to subscribers since they need to issue a GetStatus request to see the exact change anyway

Annex A
(normative)

Abstract Test Suite

A.1 Overview

This section is an abstract test suite (ATS): a compendium of test assertions applicable to server implementations of the service interface described in this standard. An ATS provides a basis for developing an executable test suite (ETS) to verify that the implementation under test (IUT) conforms to all relevant functional specifications.

The abstract test cases (assertions) are organized into indivisible test groups called conformance test classes that correspond to sets of capabilities that implementations can seek conformance to individually. Each conformance test class provides assertions checking for requirements listed in the corresponding requirements class of the main part of the document.

Server implementations seeking conformance to this standard shall pass at least the first conformance test class (A.2) and can also decide to pass other optional conformance test classes.

A.2 Conformance Test Class: Core data model for all mission types

Conformance Test Class	
http://www.opengis.net/spec/EOSPS/2.0/conf/core	
Target Type	Server Implementation
Dependency	http://www.opengis.net/spec/SPS/2.0/conf/Core
Dependency	http://www.opengis.net/spec/SPS/2.0/conf/FeasibilityController
Dependency	http://www.opengis.net/spec/SPS/2.0/conf/XMLEncoding

Conformance tests defined in this class check for requirements expressed in clause 7.1 of this standard. These tests are targeted to EO SPS server implementations.

A.2.1 An instance of the EO SPS service satisfies requirements of SPS 2.0.

Conformance Test

http://www.opengis.net/spec/EOSPS/2.0/conf/core/dependency-sps	
Requirement Req 1	An EO SPS implementation shall first pass the "Core", "Feasibility Controller" and "XML Encoding" conformance test classes of the "Sensor Planning Service Interface Standard 2.0" standard [OGC 09-000].
Test Method	Run all conformance tests defined in the "Core", "Feasibility Controller" and "XML Encoding" classes of [OGC 09-000], thus testing all SPS mandatoy operations and behavior. Also run conformance tests of optional [OGC 09-000] conformance classes supported by the service listed in the "ows:Profile" tags of the capabilities document.
Test Type	Capability

A.2.2 The DescribeTaskingResponse document contains correctly encoded EO satellite tasking parameters

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/core/tasking-params-valid	
Requirement Req 2	The *DescribeTaskingResponse* document shall contain the tasking parameters specified in this document encoded in SWE Common by using exclusively the field names, definition and units specified in the tables of this specification.
Test Method	Call the *DescribeTasking* operation on each offering listed in the service capabilities. Check that each *DescribeTaskingResponse* XML document returned by the service contains at least the mandatory tasking parameters defined in this standard. Check that all tasking parameters with a definition in the OGC namespace (i.e. with the definition attribute starting with 'http://www.opengis.net/') are well constructed. In particular, the field names, component types, definition and code space URIs and units of measure shall be the ones specified in the standard. Some of the tests are covered by the Schematron file 'eoTaskingParameters.sch' provided with the XML schemas.
Test Type	Capability

A.2.3 All vendor parameters in the DescribeTaskingResponse are optional

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/core/tasking-params-vendor	
Requirement Req 3	Additional vendor parameters shall be used only when no equivalent parameter is defined in this standard and shall be marked as optional.

Test Method	Run test A.2.2. Check that all vendor parameters (i.e. with a definition URI not listed in this specification) included in the *DescribeTaskingResponse* XML document are marked as optional by setting the *optional* attribute to "true".
Test Type	Capability

A.2.4 XML encoding of tasking parameter values is supported by the service instance

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/core/tasking-params-xml-encoding	
Requirement Req 4	An EO SPS implementation shall support the *XMLEncoding* defined in clause 8.5 of [OGC 08-094] "SWE Common Data Model 2.0 Encoding Standard" and pass the corresponding conformance tests.
Test Method	Issue a *GetCapabilities* request to the server. Check that the capabilities document contains the *XMLEncoding* listed in one of the *supportedEncoding* elements of the contents section. Check that tasking requests with parameter values encoded in XML are not rejected by the server.
Test Type	Capability

A.2.5 Tasking parameters consist of a correctly encoded instance of a subclass of *ProgrammingRequest*

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/core/tasking-params-root	
Requirement Req 5	The root of the list of tasking parameters (in *DescribeTaskingResponse*) shall be a concrete instance of the *ProgrammingRequest* class.
Test Method	Call the *DescribeTasking* operation on each offering listed in the service capabilities. Check that the "taskingParameter" element in each *DescribeTaskingResponse* XML document returned by the service contains a "swe:DataRecord" element implementing either the *SwathProgrammingRequest* class or the *CoverageProgramming--Request* class.
Test Type	Capability

A.2.6 The priority field allows for the default values even if mission specific values are also listed

Conformance Test

http://www.opengis.net/spec/EOSPS/2.0/conf/core/priority-default	
Requirement Req 6	The *Priority* field shall be of type enumeration with at least the two default priority levels 'STANDARD' and 'HIGH'.
Test Method	Call the *DescribeTasking* operation on each offering listed in the service capabilities. Check each *DescribeTaskingResponse* XML document returned by the service. If the document contains a 'Priority' field, check that at least the two options 'STANDARD' and 'HIGH are offered.
Test Type	Capability

A.2.7 A *SwathProgrammingRequest* with invalid temporal parameters is correctly rejected by the server

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/core/swath-request-time	
Requirement Req 7	An instance of the *SwathProgrammingRequest* shall either include a *TimeOfInterest* as the absolute time period of interest when provided, or set the *Cycle* attribute on each individual *SwathSegment* otherwise.
Test Method	Find out if the service has offerings supporting the *SwathProgrammingRequest* class. For each offering supporting this type of programming request (if any): - Issue *GetFeasibility* and *Submit* requests (and *Reserve* requests if supported) with no *TimeOfInterest* and no cycle numbers. Check that in all cases the service returns an exception with the code "*InvalidParameterValue*" and the locator "*taskingParameters*". - Issue *GetFeasibility* and *Submit* requests (and *Reserve* requests if supported) with both *TimeOfInterest* and cycle numbers. Check that in both cases the service returns an exception with the code "*InvalidParameterValue*" and the locator "*taskingParameters*".
Test Type	Capability

A.2.8 Expressing the time of interest as a single survey period is supported

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/core/survey-period-mandatory	
Requirement Req 8	The *TimeOfInterest* union shall not be restricted to *TimeSeries* only. A single survey period shall always be a possible option.
Test Method	Call the DescribeTasking operation on each offering listed in the service capabilities. Check that each DescribeTaskingResponse XML

	document returned by the service contains the TimeOfInterest field encoded either as a DataRecord with a single SurveyPeriod field or a DataChoice representing a choice between SurveyPeriod and TimeSeries classes.
Test Type	Capability

A.2.9 A *CoverageProgrammingRequest* containing a time series instance with inconsistent parameters is correctly rejected by the server

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/core/time-series-coherent	
Requirement Req 9	The values of the *SurveyPeriod*, *Periodicity* and *Occurrences* attributes of an instance of the *TimeSeries* class shall be such that the minimum periodicity multiplied by the number of occurrences is less than or equal to the duration of the survey period.
Test Method	Find out if the service has offerings supporting the *TimeSeries* class. For each offering supporting this type of programming request (if any), issue *GetFeasibility* a *Submit* requests (and *Reserve* requests if supported) with syntactically valid *TimeSeries* but containing inconsistent values. Check that in all cases the service returns an exception with the code *"InvalidParameterValue"* and the locator *"TimeOfInterest"*.
Test Type	Capability

A.2.10 The *FeasibilityLevel* parameter is taken into account by the server

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/core/feasibility-level-default	
Requirement Req 10	The *FeasibilityLevel* parameter shall be taken into account when it is inserted within a *GetFeasibility* request. When it is omitted the default value of 'SIMPLE' shall be assumed by the server.
Test Method	Check that the service properly takes into account the value of the *FeasibilityLevel* extension element inserted in *GetFeasibility* requests by making sure the response to a valid *GetFeasibility* request with a level of 'COMPLETE' contains more *informationUsed* elements than a request with a level of 'SIMPLE'. Check that a *GetFeasibility* request with no *FeasibiliyLevel* element returns the same result as a request with a *FeasibilityLevel* value of 'SIMPLE'.
Test Type	Capability

A.2.11 A *GetFeasibility* request containing an invalid *FeasibilityLevel* element is correctly rejected by the server

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/core/feasibility-level-valid	
Requirement Req 11	The *FeasibilityLevel* XML element shall be valid with respect to the *spsRequestExtensions.xsd* schema and inserted within an extension element of the *GetFeasibility* request.
Test Method	Issue a *GetFeasibility* request with a *FeasibilityLevel* element containing an invalid string (i.e. different than 'SIMPLE' or 'COMPLETE'). Check that the service returns an exception with the code "*InvalidParameterValue*" and the locator "*FeasibilityLevel*".
Test Type	Capability

A.2.12 The *ReferenceFeasibilityID* parameter is taken into account

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/core/ref-feasibility-id-used	
Requirement Req 12	A *Submit* or *Reserve* request containing the *ReferenceFeasibilityID* parameter shall be associated with the corresponding feasibility study.
Test Method	Inspect the status of tasks that have been submitted with a reference to a previously generated and non expired feasibility study. Verify that the acquisitions are close to what was planned in the feasibility study. Issue a warning if they are not. *Note that this test probably cannot be implemented in an automatic manner and may require a human to analyze the server response*.
Test Type	Capability

A.2.13 The *ReferenceFeasibilityID* parameter references an active feasibility study

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/core/ref-feasibility-id-correct	
Requirement Req 13	The value of *ReferenceFeasibilityID* parameter shall be the identifier of an existing and active (i.e. not expired) feasibility study report.
Test Method	Issue *Submit* requests (and *Reserve* requests if supported) with randomly generated values for the *ReferenceFeasibilityID* as well as with *ReferenceFeasibilityID* values corresponding to expired feasibility studies. Check that in all cases the service returns an exception with the code "*InvalidParameterValue*" and the locator

	"ReferenceFeasibilityID".
Test Type	Capability

A.2.14 A *Submit* or *Reserve* request with tasking parameters different than the ones used to generate the reference feasibility study is correctly rejected by the server

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/core/ref-feasibility-id-params	
Requirement Req 14	The values of tasking parameters included in a *Submit* or *Reserve* request shall be identical to the ones used in the *GetFeasibility* request referred to by the *ReferenceFeasibilityID* parameter.
Test Method	Issue *Submit* requests (and *Reserve* requests if supported) with different tasking parameters than the one used when requesting the feasibility study. Check that the service returns an exception with the code *"InvalidRequest"* and the locator *"ReferenceFeasibilityID"*.
Test Type	Capability

A.2.15 The *ReferenceFeasibilityID* parameter is valid

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/core/ref-feasibility-id-valid	
Requirement Req 15	The *ReferenceFeasibilityID* XML element shall be valid with respect to the *spsRequestExtensions.xsd* schema and inserted within an extension element of the *Submit* or *Reserve* XML request.
Test Method	Issue a *Submit* request (and a *Reserve* request if supported) with a *ReferenceFeasibilityID* element containing a sub element. Check that the service returns an exception with the code *"InvalidParameterValue"* and the locator *"ReferenceFeasibilityID"*.
Test Type	Capability

A.2.16 Successful feasibility studies include a detailed report

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/core/feasibility-study-report	
Requirement Req 16	A *StatusReport* describing the result of a feasibility study and with a request status of *'Accepted'* shall include an instance of the *FeasibilityStudy* class.
Test Method	Issue a valid *GetFeasibility* request to each offering advertised by the

	server making sure all of them lead to feasible status (e.g. requests should have a small ROI, a long period, simple level, and allow full extent on all range parameters). Check that the service always returns a *GetFeasibilityResponse* that include a *StatusReport* containing a *FeasibilityStudy* in an extension element.
Test Type	Capability

A.2.17 Grid cells and segments always have the POTENTIAL status when inserted in the report of a feasible request

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/core/feasibility-study-acc-obj-status	
Requirement Req 17	The value of the *status* attribute of *GridCell* and *Segment* instances shall be 'POTENTIAL' when these objects are used within a feasibility study report with a request status of 'Accepted' (i.e. feasible).
Test Method	Run test A.2.16. Check that all *Segment* or *GridCell* objects contained in each resulting *FeasibilityStudy* object have their *statusCode* element set to "POTENTIAL".
Test Type	Capability

A.2.18 Grid cells and segments always have the POTENTIAL or REJECTED status when inserted in the report of an unfeasible request

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/core/feasibility-study-rej-obj-status	
Requirement Req 18	The value of the *status* attribute of the *GridCell* and *Segment* instances shall be 'POTENTIAL' or 'REJECTED' when these objects are used within a feasibility study report with a request status of 'Rejected' (i.e. not feasible).
Test Method	Issue a valid *GetFeasibility* request to each offering advertised by the server and leading to unfeasible status (e.g. a request with a very large ROI, a short period, a simple feasibility level and restrictive range parameters). Check that if a *FeasibilityStudy* object is included in the response, all *Segment* or *GridCell* objects contained in it have their *statusCode* element set to "POTENTIAL" or "REJECTED", and that at least some of the cells (if cells are listed) have the value "REJECTED".
Test Type	Capability

A.2.19 The FeasibilityStudy element is valid

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/core/feasibility-study-valid	
Requirement Req 19	The *FeasibilityStudy* XML element shall be inserted in the extension slot of the SPS *StatusReport* element describing the status of a feasibility study and be valid with respect to the *eoTaskingExtensions.xsd* schema.
Test Method	Run test A.2.16. Validate the *FeasibilityStudy* element using the grammar defined in the *eoTaskingExtensions.xsd* XML schema.
Test Type	Capability

A.2.20 Status reports of accepted tasks include detailed information

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/core/prog-status	
Requirement Req 20	A *StatusReport* describing the state of a programming task and with the task status 'InExecution' shall include an instance of the *ProgrammingStatus* class.
Test Method	Issue a valid and feasible *Submit* request (e.g. a request with a small ROI, a long period, simple feasibility level and full extent for all range parameters) and wait until it is accepted by polling *GetStatus*. When the task is accepted, check that the *StatusReport* in the response includes a *ProgrammingStatus* object in an extension slot.
Test Type	Capability

A.2.21 Acquired segments are all listed in the task status report

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/core/prog-status-inc-segments	
Requirement Req 21	An instance of the *ProgrammingStatus* class shall include all segments that have been acquired (i.e. in the 'ACQUIRED' or 'VALIDATED' state) at the time the report is generated.
Test Method	Run test A.2.20. Wait until segments are acquired. Check that segments with a status of 'ACQUIRED' or 'VALIDATED' are added to the list every time the *percentCompletion* field of the *StatusReport* is incremented.
Test Type	Capability

A.2.22 Grid cells used for the feasibility study are all listed in the task status report

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/core/prog-status-inc-cells	
Requirement Req 22	An instance of the *ProgrammingStatus* class shall include the list of all cells resulting from the corresponding feasibility study (if any), with an updated status.
Test Method	Issue a valid *GetFeasibility* request to the server leading to feasible status (e.g. a request with a small ROI, a long period, simple level, and with full extent on all range parameters). If the feasibility study report contains grid cells, issue a *Submit* request with identical parameters. Wait until the status of the task changes to ACCEPTED by polling *GetStatus*. When the task is accepted, check that the *StatusReport* obtained contains the same list of grid cells as the feasibility study report.
Test Type	Capability

A.2.23 The ProgrammingStatus element is valid

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/core/prog-status-valid	
Requirement Req 23	The *ProgrammingStatus* element shall be inserted in the extension slot of the *StatusReport* element describing the status of a submitted task and be valid with respect to the *eoTaskingExtensions.xsd* schema.
Test Method	Run test A.2.20. Validate the *ProgrammingStatus* element using the grammar defined in the *eoTaskingExtensions.xsd* XML schema.
Test Type	Capability

A.2.24 GridCell attributes are consistent with the programming request

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/core/gridcell-coherent	
Requirement Req 24	The attribute values of a *GridCell* instance shall be consistent with what was requested in the original feasibility or tasking request.
Test Method	Issue a valid and feasible *GetFeasibility* request. Check several things about the grid cells listed in the response: - Check that the footprint of each cell intersects the region of interest specified in the request. - If the cell is in the 'POTENTIAL' state, check that the dates in the *nextAttemptDate*, *lastAttemptDate* and *estimatedCompletionDate*

	fields (if present) are within the survey period specified in the request Run test A.2.21. Issue a *GetStatus* request on the task to obtain a status report. Check the above two criteria again on the grid cells included in the response.
Test Type	Capability

A.2.25 The CRS of the footprint polygon is EPSG 4326

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/core/gridcell-footprint-srs	
Requirement Req 25	The value of the *srsName* attribute of the footprint polygon shall be "http://www.opengis.net/def/crs/EPSG/0/4326".
Test Method	Run test A.2.24. Check that the footprint polygon of all cell objects listed in the returned status reports has a *srsName* attribute with the value "http://www.opengis.net/def/crs/EPSG/0/4326".
Test Type	Capability

A.2.26 The footprint polygon does not contain holes and uses the posList syntax

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/core/gridcell-footprint-syntax	
Requirement Req 26	The footprint polygon shall contain only an exterior ring expressed as a *LinearRing* element which coordinates shall be encapsulated in a *posList* element.
Test Method	Run test A.2.24. Check that the footprint polygon of all cell objects listed in the status reports generated by the service use the constrained syntax
Test Type	Capability

A.2.27 Segment attributes are consistent with the programming request

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/core/segment-coherent	
Requirement Req 27	The attribute values of a *Segment* instance shall be consistent with what was requested in the original feasibility or tasking request.
Test Method	Issue a valid and feasible *GetFeasibility* request. Check several things about the segments listed in the response:

	- Check that the footprint of each segment intersects the region of interest specified in the request. - Check that the start and stop acquisition dates of each segment are within the survey period specified in the request. - Check that the segment acquisition angles (incidence, pitch, roll, yaw specified within the *EarthObservationEquipment* object) are within the range requested. Check that the instrument, platform and mode used to acquire the segment (as specified in the *EarthObservationEquipment* object) match the ones requested. Run test A.2.21. Issue a *GetStatus* request on the task to obtain a status report. Check the above criteria again on all segments included in the report.
Test Type	Capability

A.2.28 Segment footprint polygons satisfy the same constraints as grid cell footprint polygons

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/core/segment-footprint-syntax	
Requirement Req 28	The polygon element used to describe the segment's footprint shall comply with Req 25 and Req 26.
Test Method	Run tests A.2.25 and A.2.26, applying the checks to segments rather grid cells.
Test Type	Capability

A.2.29 Unit of measures are all SI units

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/core/segment-units	
Requirement Req 29	The unit of measure of all angular quantities used within the *EarthObservationEquipment* element shall be decimal degree ('deg' in UCUM syntax) and the unit of distance shall be meter ('m' in UCUM syntax).
Test Method	Run test A.2.27. Check that in all *EarthObservationEquipment* objects generated by the service the proper units are used.
Test Type	Capability

A.2.30 Latest status report contains all acquired segments

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/core/get-status-latest	
Requirement Req 30	The response to a *GetStatus* request with no '*since*' parameter shall consist of a single status report containing all segments acquired since the task was submitted.
Test Method	Run test A.2.21. Issue a *GetStatus* request with no *since* parameter. Check that the response consists of a single status reports containing a list of segments. Check that all segments in the list with a status of 'ACQUIRED' or 'VALIDATED' have an acquisition date that is less than the current date.
Test Type	Capability

A.2.31 Contents of status report are filtered according to since parameter

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/core/get-status-since	
Requirement Req 31	The response to a *GetStatus* request with a '*since*' parameter shall contain a status report with only segments and grid cell objects whose status has changed after the date given in the *since* parameter.
Test Method	Run test A.2.21. Issue a *GetStatus* request with a *since* parameter greater than the acquisition date of one of the acquired segments. Check that only segments acquired after this date are included in the resulting status report.
Test Type	Capability

A.3 Conformance Test Class: Additional extensions for optical systems

Conformance Test Class	
http://www.opengis.net/spec/EOSPS/2.0/conf/opt	
Target Type	Server Implementation
Dependency	http://www.opengis.net/spec/EOSPS/2.0/conf/core

A.3.1 An implementation supporting the optical extension is conformant with the core of this standard

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/opt/dependency-core	
Requirement Req 32	An EO SPS implementation passing the "Additional extensions for optical missions" conformance class shall first pass the core conformance class.
Test Method	Apply all tests from conformance test class A.1 to the service implementation supporting the extension.
Test Type	Capability

A.3.2 An implementation supporting the optical extension uses the OPT tasking parameters defined in this standard.

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/opt/tasking-params-used	
Requirement Req 33	The EO SPS implementation shall use one of *AcquisitionParametersOPT* or *ValidationParametersOPT* class in its *DescribeTaskingResponse*.
Test Method	Call the *DescribeTasking* operation on each offering listed in the service capabilities. Check that at least one of the *DescribeTaskingResponse* XML document returned by the service contain an instance of *AcquisitionParametersOPT* or *ValidationParametersOPT*.
Test Type	Capability

A.3.3 A custom instrument mode code space is URL accessible and GML encoded

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/core/tasking-params-codespace-gml	
Requirement Req 34	A custom optical instrument code space shall be URL accessible and encoded in GML 3.2 format. The code space document shall pass the corresponding conformance test class of [OGC 07-036].
Test Method	Run test A.3.2. For each *DescribeTaskingResponse* XML document returned by the service, if it contains the *InstrumentMode* parameter with a custom code space (i.e. different from the default one define in this standard, see 7.2.2.1), connect to the URL of the code space and check that the document returned is a GML dictionary. Apply tests defined in GML 3.2.1 conformance test suite to the code space document.
Test Type	Capability

A.3.4 A custom instrument mode code space includes the default spectral modes

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/opt/tasking-params-codespace-default	
Requirement Req 35	If a custom instrument mode code space is defined, it shall contain the two default options 'MULTISPECTRAL' and 'PANCHROMATIC'.
Test Method	Run test A.3.3. Check that the GML dictionary document received contains two dictionary entries with the name 'MULTISPECTRAL' and 'PANCHROMATIC'.
Test Type	Capability

A.4 Conformance Test Class: Additional extensions for SAR systems

Conformance Test Class	
http://www.opengis.net/spec/EOSPS/2.0/conf/sar	
Target Type	Server Implementation
Dependency	http://www.opengis.net/spec/EOSPS/2.0/conf/core

A.4.1 An implementation supporting the SAR extension is conformant with the core of this standard

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/sar/dependency-core	
Requirement Req 36	An EO SPS implementation passing the "Additional extensions for SAR missions" conformance class shall first pass the core conformance class.
Test Method	Apply all tests from conformance test class A.1 to the service implementation supporting the extension.
Test Type	Capability

A.4.2 An implementation supporting the SAR extension uses the SAR tasking parameters defined in this standard.

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/sar/tasking-params-used	
Requirement Req 37	The EO SPS implementation shall use one of *AcquisitionParametersSAR* or *ValidationParametersSAR* class in its *DescribeTaskingResponse*.
Test Method	Call the *DescribeTasking* operation on each offering listed in the service capabilities. Check that at least one of the *DescribeTaskingResponse* XML document returned by the service contain an instance of *AcquisitionParametersSAR* or *ValidationParametersSAR*. If it is the case, check that the structure of the parameter conforms to the structure defined in the SAR requirements class.
Test Type	Capability

A.4.3 A code space for SAR instrument modes is defined

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/sar/tasking-params-codespace	
Requirement Req 38	A code space identifying and defining the supported SAR instrument modes shall be provided.
Test Method	For each *DescribeTaskingResponse* XML document returned by the service, if it contains the *InstrumentMode* parameter, check that a code space is listed in the form of a URL.
Test Type	Capability

A.4.4 A custom SAR mode code space is URL accessible and GML encoded

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/sar/tasking-params-codespace-gml	
Requirement Req 39	A custom SAR code space shall be URL accessible and encoded in GML 3.2 format. The code space document shall pass the corresponding conformance test class of [OGC 07-036].
Test Method	Run test A.4.3. Connect to the URL identifying the code space and check that the document received is a GML dictionary. Apply tests defined in GML 3.2.1 conformance test suite to the document.
Test Type	Capability

A.5 Conformance Test Class: GetSensorAvailability operation

Conformance Test Class	
http://www.opengis.net/spec/EOSPS/2.0/conf/gsa	
Target Type	Server Implementation
Dependency	http://www.opengis.net/spec/EOSPS/2.0/conf/core

A.5.1 The *GetSensorAvailability* request is listed in the service capabilities

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/gsa/capabilities	
Requirement Req 40	The *GetSensorAvailability* operation shall be listed in the capabilities document of the service.
Test Method	Issue a *GetCapabilities* request to the service. Verify that the *GetSensorAvailability* operation is listed in the *OperationsMetadata* section of the capabilities document.
Test Type	Capability

A.5.2 An invalid *GetSensorAvailability* request is correctly rejected by the server

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/gsa/request-valid	
Requirement Req 41	The XML message encoding a *GetSensorAvailability* request shall be valid with respect to the getSensorAvailability.xsd XML schema. The server shall correctly reject an invalid request by returning the proper correctly formatted exception as defined in [OGC 06-121] and [OGC 09-001].
Test Method	Issue an invalid *GetSensorAvailability* request to the service (i.e. with a content that does not respect the grammar defined in the *getSensorAvailability.xsd* XML schema). Verify that an *InvalidRequest* exception is returned.
Test Type	Capability

A.5.3 A *GetSensorAvailability* response is valid with respect to the XML schema

Conformance Test
http://www.opengis.net/spec/EOSPS/2.0/conf/gsa/response-valid

Requirement Req 42	The XML message encoding a *GetSensorAvailabilityResponse* shall be valid with respect to the getSensorAvailability.xsd XML schema.
Test Purpose	Verify that .
Test Method	Issue valid *GetSensorAvailability* requests to the server using the various sensor identifiers advertised in its capabilities. Validate each *GetSensorAvailabilityResponse* document using the grammar defined in the *getSensorAvailability.xsd* XML schema.
Test Type	Capability

A.5.4 An exception is returned when the sensor identifier is invalid

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/gsa/sensor-id-valid	
Requirement Req 43	The *sensor* parameter of a *GetSensorAvailability* request shall contain the identifier of a sensor listed in the capabilities. If this is not the case, the server shall return an exception with the code "InvalidParameterValue" as specified in [OGC 06-121].
Test Method	Issue a *GetSensorAvailability* request to the server using a sensor identifier that is not advertised in its capabilities. The request is otherwise valid. Check that the service returns an exception with the code "*InvalidParameterValue*" and the locator "*SensorIdentifier*".
Test Type	Capability

A.6 Conformance Test Class: Validate operation

Conformance Test Class	
http://www.opengis.net/spec/EOSPS/2.0/conf/val	
Target Type	Server Implementation
Dependency	http://www.opengis.net/spec/EOSPS/2.0/conf/core

A.6.1 The Validate request is listed in the service capabilities

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/val/capabilities	
Requirement Req 44	The *Validate* operation shall be listed in the capabilities document of the service.
Test Method	Issue a *GetCapabilities* request to the service. Verify that the *Validate* operation is listed in the *OperationsMetadata* section of the capabilities document.
Test Type	Capability

A.6.2 An invalid Validate request is correctly rejected by the server

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/val/request-valid	
Requirement Req 45	The XML message encoding a *Validate* request shall be valid with respect to the validate.xsd XML schema. The server shall correctly reject an invalid request by returning the proper correctly formatted exception as defined in [OGC 06-121] and [OGC 09-001].
Test Method	Issue an invalid *Validate* requests to the service (i.e. with a content that does not respect the grammar defined in the *validate.xsd* XML schema). Verify that an *InvalidRequest* exception is returned.
Test Type	Capability

A.6.3 A Validate response is valid with respect to the XML schema

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/val/response-valid	
Requirement	The XML message encoding a *ValidateResponse* shall be valid with

Req 46	respect to the validate.xsd XML schema.
Test Method	Run test A.2.20. Wait for some segments to be acquired. Issue valid *Validate* requests to the server using the task and segment identifiers obtained from the accepted task. Validate each *ValidateResponse* document using the grammar defined in the *validate.xsd* schema.
Test Type	Capability

A.6.4 A Validate request with an invalid task identifier is correctly rejected by the server

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/val/task-id-valid	
Requirement Req 47	The *task* parameter of a *Validate* request shall contain the identifier of an existing task. If this is not the case, the server shall return an exception with the code "InvalidParameterValue" as specified in [OGC 06-121].
Test Method	Run test A.2.20. Issue a *Validate* request to the server using a a randomly generated task identifier but otherwise valid. Check that the service returns an exception with the code "*InvalidParameterValue*" and the locator "*TaskIdentifier*".
Test Type	Capability

A.6.5 A *Validate* request with an invalid segment identifier is correctly rejected by the server

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/val/segment-id-valid	
Requirement Req 48	The *segmentID* parameter of a *Validate* request shall contain the identifier of an acquired segment listed in the status report of the selected task. If this is not the case, the server shall return an exception with the code "InvalidParameterValue" as specified in [OGC 06-121].
Test Method	Run test A.2.20. Wait for some segments to be acquired. Issue a *Validate* request to the server using segment identifiers different from the ones listed in the *StatusReport* but otherwise valid. Check that the service returns an exception with the code "*InvalidParameterValue*" and the locator "*SegmentIdentifier*".
Test Type	Capability

A.6.6 A *Submit* request with an invalid *ManualValidation* element is correctly rejected by the server

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/val/manual-validation-valid	
Requirement Req 49	The *ManualValidation* XML element shall be valid with respect to the spsRequestExtensions.xsd schema and inserted within an extension element of the Submit or Reserve XML request.
Test Method	Issue a *Submit* request (and *Reserve* request if supported) with a *ManualValidation* element containing a value other than 'true' or 'false'. Check that the service returns an exception with the code "*InvalidParameterValue*" and the locator "*ManualValidation*".

A.7 Conformance Test Class: SubmitSegmentByID operation

Conformance Test Class	
http://www.opengis.net/spec/EOSPS/2.0/conf/sid	
Target Type	Server Implementation
Dependency	http://www.opengis.net/spec/EOSPS/2.0/conf/core

A.7.1 The *SubmitSegmentByID* request is listed in the service capabilities

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/sid/capabilities	
Requirement Req 50	The *SubmitSegmentByID* operation shall be listed in the capabilities document of the service.
Test Method	Issue a *GetCapabilities* request to the service. Verify that the *SubmitSegmentByID* operation is listed in the *OperationsMetadata* section of the capabilities document.
Test Type	Capability

A.7.2 An invalid *SubmitSegmentByID* request is correctly rejected by the server

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/sid/response-valid	
Requirement Req 51	The response to the *SubmitSegmentByID* operation shall be an instance of the *SubmitResponse* class defined in [OGC 09-000].
Test Method	Run test A.2.16. Issue valid *SubmitSegmentByID* requests to the service using the task and segment identifiers obtained in the feasibility response. Check that the service returns a *SubmitResponse* object in all case and validate it with the *spsSubmit.xsd* XML schema defined in [OGC 09-000].
Test Type	Capability

A.7.3 A SubmitSegmentByID response is valid with respect to the XML schema

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/sid/request-valid	
Requirement	The XML message encoding a *SubmitSegmentByID* request shall be

Req 52	valid with respect to the submitSegmentByID.xsd XML schema. The server shall correctly reject an invalid request by returning the proper correctly formatted exception as defined in [OGC 06-121] and [OGC 09-001].
Test Method	Issue various invalid *SubmitSegmentByID* requests to the service (i.e. with a content that does not respect the grammar defined in the *submitSegmentByID.xsd* XML schema). Verify that an *InvalidRequest* exception is returned in each case.
Test Type	Capability

A.7.4 A SubmitSegmentByID request with an invalid task identifier is correctly rejected by the server

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/sid/task-id-valid	
Requirement Req 53	The *task* parameter of a *SubmitSegmentByID* request shall contain the identifier of an existing and active feasibility study. If this is not the case, the server shall return an exception with the code "InvalidParameterValue" as specified in [OGC 06-121].
Test Method	Run test A.2.16. Issue a *SubmitSegmentByID* request to the service using a randomly generated task identifier but otherwise valid. Check that the service returns an exception with the code *"InvalidParameterValue"* and the locator *"TaskIdentifier"*.
Test Type	Capability

A.7.5 A SubmitSegmentByID request with an invalid segment identifier is correctly rejected by the server

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/sid/segment-id-valid	
Requirement Req 54	The *segmentID* parameter of a *SubmitSegmentByID* request shall contain the identifier of a feasible segment listed in the status report of the selected feasibility study. If this is not the case, the server shall return an exception with the code "InvalidParameterValue" as specified in [OGC 06-121].
Test Method	Run test A.2.16. Issue a *SubmitSegmentByID* request to the service using randomly generated segment identifiers but otherwise valid. Check that the service returns an exception with the code *"InvalidParameterValue"* and the locator *"SegmentIdentifier"*.
Test Type	Capability

A.8 Conformance Test Class: SOAP binding

Conformance Test Class	
http://www.opengis.net/spec/EOSPS/2.0/conf/soap	
Target Type	Server Implementation
Dependency	http://www.opengis.net/spec/EOSPS/2.0/conf/core
Dependency	http://www.opengis.net/spec/SPS/2.0/conf/SOAP

A.8.1 The SOAP binding is implemented for all operations

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/soap/all-operations	
Requirement Req 55	The EO SPS implementation shall allow access to all supported operations via the SOAP binding defined in [OGC 09-000].
Test Method	Send valid SOAP requests to call each operation supported by the service. Verify that the server never returns an exception.
Test Type	Capability

A.8.2 The SOAP binding is implemented as prescribed in [OGC 09-000]

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/soap/dependency-sps	
Requirement Req 56	An EO SPS implementation passing the "SOAP binding" conformance class shall first pass the "SOAP" conformance class of [OGC 09-000].
Test Purpose	Verify that .
Test Method	Issue SOAP requests with invalid SOAP Action URIs to all operation advertised by the server. Check that the service returns a SOAP fault in all cases. Issue valid SOAP requests to all operation advertised by the server and check that responses are not exceptions and contain the right SOAP Action URI.
Test Type	Capability

A.8.3 The proper SOAP Action URIs are used

Conformance Test

http://www.opengis.net/spec/EOSPS/2.0/conf/soap/action-uris	
Requirement Req 57	The proper SOAP action URI shall be used when calling EO SPS operations via the SOAP binding (version 1.1 or 1.2).
Test Method	Issue SOAP requests for the *GetSensorAvailability*, *Validate* and *SubmitSegmentByID* operations with valid SOAP action URIs. Check that the server returns a valid *GetSensorAvailabilityResponse*, *ValidateResponse*, and *SubmitResponse* respectively encapsulated in a SOAP message. Issue SOAP requests for the *GetSensorAvailability*, *Validate* and *SubmitSegmentByID* operations with invalid SOAP action URIs. Check that the server returns an exception encapsulated in a SOAP fault. Check that the format of the SOAP fault is as specified by [OGC 09-000].
Test Type	Capability

A.8.4 The *GetFeasibility* operation can be called asynchronously

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/soap/gf-wsa-support	
Requirement Req 58	The EO SPS implementation shall support asynchronous access to the *GetFeasibility* operation by using WS-Addressing *ReplyTo* mechanism.
Test Method	Issue a SOAP *GetFeasibility* request with a correct WS-Addressing header containing a *ReplyTo* URL. Verify that the service does not return an exception and that a *GetFeasibilityResponse* message is sent back to the specified URL.
Test Type	Capability

A.8.5 The *GetFeasibility* operation called with WSA header returns a single response message

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/soap/gf-wsa-single-response	
Requirement Req 59	When the header of the SOAP *GetFeasibility* request contains the ReplyTo element, a single response shall be sent to the *ReplyTo* URL.
Test Method	Run test A.8.4. Check that the *GetFeasiblilityResponse* received at the *ReplyTo* URL contains the *FeasibilityResult* element and has the "ACCEPTED" or "REJECTED" status.
Test Type	Capability

A.9 Conformance Test Class: Notifications

Conformance Test Class	
http://www.opengis.net/spec/EOSPS/2.0/conf/notif	
Target Type	Server Implementation
Dependency	http://www.opengis.net/spec/EOSPS/2.0/conf/core
Dependency	http://www.opengis.net/spec/SPS/2.0/conf/ChannelBasedPubSub

A.9.1 Notifications are implemented as defined in SPS 2.0

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/notif/dependency-sps	
Requirement Req 60	An EO SPS implementation passing the "Notifications" conformance test class shall first pass the "Channel Based PubSub" conformance test class defined in the Sensor Planning Service 2.0 Interface Standard [OGC 09-000].
Test Method	Apply all conformance tests defined in clause 11.8 of [OGC 09-000].

A.9.2 Basic notification topics are supported

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/notif/events-support-basic	
Requirement Req 61	The EO SPS implementation shall advertise and generate notifications for events *TaskRequestAccepted*, *TaskRequestRejected*, *TaskSubmitted*, *TaskCompleted*, *TaskFailed and DataPublished*.
Test Method	Issue a *GetCapabilities* request and check that the *TaskSubmitted*, *TaskCompleted*, *TaskFailed* and *DataPublished* notification topics are listed in the *NotificationProducerMetadata* section of the capabilities document.
Test Type	Capability

A.9.3 Reservation notifications are supported

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/notif/events-support-reserve	
Requirement	An EO SPS implementation supporting task reservation shall

Req 62	advertise and generate notifications for the following events: *TaskReserved, TaskConfirmed and ReservationExpired.*
Test Method	Issue a *GetCapabilities* request to the server. Check if the *Reserve* operation is listed in the *OperationMetadata* section of the capabilities document. If it is listed, verify that the *TaskReserved, TaskConfirmed and ReservationExpired* topics are listed in the *NotificationProducerMetadata* section.
Test Type	Capability

A.9.4 Cancellation notifications are supported

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/notif/events-support-cancel	
Requirement Req 63	An EO SPS implementation supporting task cancellation shall advertise and generate notifications for the event *TaskCancelled.*
Test Method	Issue a *GetCapabilities* request to the server. Check if the *Cancel* operation is listed in the *OperationMetadata* section of the capabilities document. If it is listed, verify that the *TaskCancelled* topic is listed in the *NotificationProducerMetadata* section.
Test Type	Capability

A.9.5 Update notifications are supported

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/notif/events-support-update	
Requirement Req 64	An EO SPS implementation supporting task updates shall advertise and generate notifications for the event *TaskUpdated.*
Test Method	Issue a *GetCapabilities* request to the server. Check if the *Update* operation is listed in the *OperationMetadata* section of the capabilities document. If it is listed, verify that the *TaskUpdated* topic is listed in the *NotificationProducerMetadata* section.
Test Type	Capability

A.9.6 EO satellite specific notifications are supported

Conformance Test	
http://www.opengis.net/spec/EOSPS/2.0/conf/notif/events-support-eo	
Requirement Req 65	The EO SPS implementation shall advertise and generate notifications for the events *SegmentPlanned, SegmentAcquired,*

	SegmentValidated, *SegmentCancelled* and *SegmentFailed*.
Test Method	Issue a *GetCapabilities* request and check that the *SegmentPlanned*, *SegmentAcquired*, *SegmentValidated*, *SegmentCancelled* and *SegmentFailed* notification topics are listed in the *NotificationProducerMetadata* section of the capabilities document. Submit a new task to the server and wait until it is accepted. Issue a *Subscribe* request to each of the notification topics listed above. Verify that the notification messages received at the specified endpoint are generated only when the corresponding segment status transition occurs. This can be done by issuing a *GetStatus* request every time a notification is received and making sure that at least one of the segments listed in the returned status report had its status updated to the expected code.

Annex B
(normative)

XML Schema documents

In addition to this document, this standard includes several normative XML Schema documents that are bundled in a zip file with the present document.

After OGC acceptance of version 2.0.0 of this standard, these XML Schema documents will be posted online at the URL http://schemas.opengis.net/eosps/2.0. In the event of a discrepancy between the bundled and online versions of the XML Schema documents, the online files shall be considered authoritative.

The requirements classes specified in this document use six specified XML Schema documents included in the zip file with this document. These XML Schema documents combine the XML schema fragments listed in various sub-clauses of this document, eliminating duplications. These XML Schema documents match the UML packages described in the main section of the document and are named:

eosps.xsd

spsRequestsExtensions.xsd

eoTaskingExtensions.xsd

getSensorAvailability.xsd

submitSegmentByID.xsd

validate.xsd

These XML Schema documents use and build on the Sensor Planning Service 2.0, OWS Common 2.0, the Earth Observation Metadata profile of Observations & Measurements and the Geographic Markup Language 3.2.1 XML Schema documents specified in [OGC 09-000], [OGC 06-121], [OGC 10-157] and [OGC 07-036] respectively.

All these XML Schema documents contain documentation of the meaning of each element and attribute, and this documentation shall be considered normative.

Annex C
(informative)

Example XML documents

C.1 SPS request and response examples

C.1.1 GetCapabilities GET request example

http://www.domain.com/services/SPS?service=SPS&request=GetCapabilities

C.1.2 GetCapabilities response example

http://schemas.opengis.net/eosps/2.0/examples/GetCapabilitiesResponse.xml

C.1.3 DescribeTasking request example

http://schemas.opengis.net/eosps/2.0/examples/DescribeTaskingRequest.xml

C.1.4 DescribeTasking response example

http://schemas.opengis.net/eosps/2.0/examples/DescribeTaskingResponse_OPT.xml
http://schemas.opengis.net/eosps/2.0/examples/DescribeTaskingResponse_SAR.xml

C.1.5 GetFeasibility request example

http://schemas.opengis.net/eosps/2.0/examples/GetFeasibilityRequest.xml

C.1.6 GetFeasibility response example

http://schemas.opengis.net/eosps/2.0/examples/GetFeasibilityResponseOPT.xml
http://schemas.opengis.net/eosps/2.0/examples/GetFeasibilityResponseSAR.xml

C.1.7 Submit request example

http://schemas.opengis.net/eosps/2.0/examples/SubmitRequest.xml

C.1.8 Submit response example

http://schemas.opengis.net/eosps/2.0/examples/SubmitResponse.xml

C.1.9 GetStatus request example

http://schemas.opengis.net/eosps/2.0/examples/GetStatusRequest.xml

C.1.10 GetStatus response example

http://schemas.opengis.net/eosps/2.0/examples/GetStatusResponse.xml

C.2 WS-Notification examples

C.2.1 Subscribe request example

http://schemas.opengis.net/eosps/2.0/examples/SubscribeRequest.xml

C.2.2 Notification message examples

http://schemas.opengis.net/eosps/2.0/examples/NotifySegmentAcquired.xml

C.3 GML code space examples

C.3.1 Instrument mode code space example (OPT)

http://schemas.opengis.net/eosps/2.0/examples/SPOTInstrumentModes.xml

C.3.2 Instrument mode code space example (SAR)

http://schemas.opengis.net/eosps/2.0/examples/TerraSARXInstrumentModes.xml

Annex D : Revision History Revision History

Date	Release	Editor(s)	Primary clauses modified	Description
2006-07-28	0.0.1	Didier Giacobbo	Initial version	Initial version;
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
2007-01-16	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, SWE Common instance
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	Minor update	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)
2007-05-07	0.9.4	Philippe Mérigot		Editing corrections SOAP version 1.1 supported Input parameters : parameter Priority § 21 (Implementation guidance) has been replaced by § 20 Multi provider scenario Schemas: recursivity in Input parameters (§ 8.3), GML version (v3.1.1) and target namespace Scenarios modified

				New request response examples in Annex B
2007-10-08	0.9.5	Philippe Mérigot	Major update	- Reference to UM ICD - OWS version 1.1.0 - Unit of measure: UCUM codes recommended - § 8 and 9 inverted and modified - *(IMPR#81)* ProgrammingRequestMode parameter harmonised between SAR and OHR Operation - GetFeasibility : feasibility levels - Cancel : asynchronous - DescribeSensor response aligned with SOS - Shared requests and responses acknowledgement (GetFeasibility, Submit, Cancel) - Progress reports - Soap messages examples (annex B)
2009-11-19	2.0 draft	Philippe Mérigot Alexandre Robin	Major update	- Based on OGC SPS v2.0 specification and sweCommon v2.0 specification - Tasking parameters reviewed - Use of sweCommon for sending values simplified - Status codes and notification topics
2010-06-23	2.0 draft	Alexandre Robin	Major Update	- Updated UML models to match OGC practice - Reorganization into requirements classes - Improved ATS with mapping to requirements - Use of EO GML (GML 3.2) for feasibility study and status report content - Added FeasibilityLevel, FeasibilityID in Submit request - Added expiration date in Feasibility Study - Added SubmitSegmentByID operation
2010-11-25	2.0	Alexandre Robin	Minor	- Minor edits for final submission - Updated references to SPS 2.0 clauses and URLs of conformance classes - Updated URLs of some SWE definitions to match already registered GM_* geometries.

