Open Geospatial Consortium Inc.

Date: 2007-08-02

Reference number of this document: OGC® **07-014r3**

Version: 1.0

Category: OpenGIS® Implementation Specification

Editor: Ingo Simonis, University of Muenster

Change Request Editor: Phillip C. Dibner, Ecosystem Associates

OpenGIS® Sensor Planning Service Implementation Specification

Notice

This document is an official, member approved, OGC Standard.

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

Document type:	OpenGIS® Implementation Specification
Document subtype:	Interface
Document stage:	Adopted
Document language:	English

Contents

Page

Figures

Page

i. Preface

The OpenGIS Sensor Planning Service (SPS) Implementation Specification is designed to enable an interoperable service by which a client can determine collection feasibility for a desired set of collection requests for one or more sensors/platforms, or a client may submit collection requests directly to these sensors/platforms. Specifically, the document specifies interfaces for requesting information describing the capabilities of a SPS for determining the feasibility of an intended sensor planning request, for submitting such a request, for inquiring about the status of such a request, for updating or cancelling such a request, and for requesting information about further OGC Web services that provide access to the data collected by the requested task.

Suggested additions, changes, and comments on this 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.

The changes made in this document version, relative to the previous version, are tracked by Microsoft Word, and can be viewed if desired. If you choose to submit suggested changes by editing this document, please first accept all the current changes, and then make your suggested changes with change tracking on.

ii. Document terms and definitions

This document uses the specification terms defined in Subclause 5.3 of [OGC 05-008], which is based on the ISO/IEC Directives, Part 2, Rules for the structure and drafting of International Standards. In particular, the word “shall” (not “must”) is the verb form used to indicate a requirement to be strictly followed to conform to this specification

iii. Submitting organizations

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

University of Muenster

iv. Document contributor contact points

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

Name	Organization
Alexander Walkowksi	University of Muenster
Alexandre Robin	UAH
Ingo Simonis (editor)	University of Muenster
Jeff Lansing	Polexis, Inc.
Jim Greenwood	SeiCorp, Inc
Johannes Echterhoff	University of Muenster
John Davidson	Image Matters
Mark Priest	3eti
Mike Botts	UAH
Phillip Dibner (CR Editor)	Ecosystem Associates
Simon Cox	CSIRO

v. Revision history

Date	Release	Editor	Primary clauses modified	Description
2003-01-14	0.0.6	Jeff Lansing	initial version	initial version; deprecated
2005-10-20	0.0.30	Ingo Simonis	new version	0.0.6 deprecated. All operations and information model changed.
2005-12-02	r1	Ingo Simonis	all	adapted to schema changes
2006-02-06	r2	Johannes Echterhoff	all	adapted to final schema changes. Examples new. Version intended to be part of RFC.
2006-02-09	r2	Phillip Dibner	Future Work	content added
2006-02-12	r2	Ingo Simonis	all	latest adaptions.
2007-02-02	0.1.5 of new version, 07-014	Phillip Dibner	all	updates to schema and material that depends on it, for compatibility with sweCommon 1.0; also changes resulting from comments to the RFC.
2007-08-02	R3	Carl Reed	various	Ready for posting as an OGC Standard

vi. Changes to the OGC Abstract Specification

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

vii. Future work

This SPS specification provides information concerning content and encoding of the parameter data that has to be provided in order to task a sensor. Though it defines mechanisms to restrict possible instance documents, e.g. the integer data of the parameter *speed* has to be within the interval [0,100], there is currently no mechanism to restrict parameters or the possible parameter values in dependence on other parameters. Example: You must deliver the parameter A if parameter B is greater than 50 and parameter C is before ten days from now'. Such a feature will remain for future versions.

Future work must also incorporate two notions that are not included in the current revision of this document. Neither of these issues applies solely to sensor systems, and the solutions may lie partly or wholly within other domains defined by the OGC architecture. Nonetheless, given their relevance to sensors, we discuss them here.

Volatility of Service Descriptions

The first issue is volatility of descriptive metadata that applies to the sensor, to its platform, or to its area of service. Many sensors can assume a variety of configurations, and the same is true of sensor platforms. An SPS Capabilities document, possibly combined with a SensorML document, might provide a detailed description of a sensing device or its platform at one moment that is not valid at another time. If such documents were saved in an online catalogue, and retrieved when it was no longer valid, users might find that they are unsuccessful in requesting the acquisition of data that the service description documents claim is available. Worse yet, the user might be able to request a collection that doesn't conform to the advertised specifications, leading to wasted expenses or incorrect interpretation. The problem could be even more severe, or even dangerous, in the case of a reconfigurable airborne platform.

The issue of volatile configuration is not limited to sensor systems. Data services like WMS, WFS, and WCS may be quite dynamic. The area they cover and the datatypes they provide may change with time. There needs to be a mechanism that prevents this type of confusion, either by preventing volatile information from being cached by OGC catalogs, or by including some metadata in the service description documents that tells whether or for how long the documents are valid.

Clearly, this is a generic issue for the OGC architecture. However, it is particularly relevant to sensors and the sensor web, especially where configurable or mobile sensors are concerned. It must be addressed, as a matter of high priority, in future work.

Role-Based Authentication and Control (RBAC)

The second requirement involves the very real need in practical systems for a clear definition of the roles played by different users of the sensor system, and the corresponding permissions that accrue to each of them. The individual who wishes to acquire data from a sensor is not necessarily the same person who configures or enables the sensor to operate in a particular mode.

For example, a multispectral scanner may have the capability to collect radiometric data from ten or more bands in the electromagnetic spectrum, but the image processing system or data transmission circuitry associated with that scanner may only have the capacity to accommodate three of those bands at a single time. In general, a request to collect an image using three particular bands will not automatically reconfigure the scanner package so that those three bands are routed to the processing or data transmission subsystems. This task must be done by someone who has access to and familiarity with the sensor configuration controls, not by the person who has requested the image.

A more dramatic example applies to sensor payloads carried by occupied or unoccupied airborne systems. The individuals who request imagery from a certain location on the earth's surface, or who configure the sensor payload, are likely not the same people who fly the aircraft, or even who develop and file the aircraft's flight plan.

These examples point to the need for Role-Based Authentication and Configuration (RBAC) in the operational management of sensing devices and control systems. Some of this technology might arise from the Geographic Digital Rights Management (GeoDRM) experimentation that was performed concurrently with this specification. Whether or not this turns out to be the case, future work on the SPS must address RBAC for sensor systems.

Foreword

The Sensor Planning Service is part of the OGC Sensor Web Enablement document suite.

This document includes four annexes; Annexes A and B are normative, and Annexes C and D are 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 specification set forth in this document.

Introduction

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

If you want get a quick overview of SPS in conjunction with some additional informative notes, please consider reading section 20, *SPS running example*, first.

OpenGIS® Sensor Planning Service Implementation Specification

1 Scope

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

2 Conformance

Conformance with this specification shall be checked using all the relevant tests specified in Annex A (normative).

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.

ISO 19105:2000, *Geographic information — Conformance and Testing*

OGC 05-008, *OpenGIS® Web Services Common Specification*

This OWS Common Specification contains a list of normative references that are also applicable to this Implementation Specification.

OGC 05-114 *OpenGIS®* Web Notification Service

OGC 05-090 SWE Architecture

OGC 05-088 SOS

OGC 05-087 O&M

OGC 05-086 SensorML

OGC 04-092r4 GML

In addition to this document, this specification includes several normative XML Schema Document files as specified in Annex B.

4 Terms and definitions

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

4.1 asset

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

4.2 asset management system

synonyms: acquisition system, asset support system
a system for controlling the effective utilization of an asset

4.3 collection

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

4.4 requirement

something that is necessary in advance

5 Conventions

5.1 Abbreviated terms

Most of the abbreviated terms listed in Subclause 5.1 of the OWS Common Implementation Specification [OGC 05-008] apply to this document, plus the following abbreviated terms.

SOS	Sensor Observation Service
WNS	Web Notification Service
SAS	Sensor Alert Service
SWE	Sensor Web Enablement
O&M	Observation and Measurement
SensorML	Sensor Model Language

TML Transducer Markup Language

5.2 UML notation

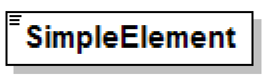
Most diagrams that appear in this specification are presented using the Unified Modeling Language (UML) static structure diagram, as described in Subclause 5.2 of [OGC 05-008].

5.3 XMLSpy notation

Most diagrams that appear in this specification are presented using an XML schema notation defined by the XMLSpy[1] product and described in this subclause. XML schema diagrams are for informative use only though they shall reflect the accompanied UML and schema perfectly.

5.3.1 Element

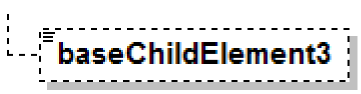
A named rectangle representing the most basic part of the XML Schema notation. Each represents an XML "Element" token. Each Element symbol can be elaborated with extra information as shown in the examples below.



This is a mandatory simple element. Note the upper left corner of the rectangle indicates that data is contained in this element.

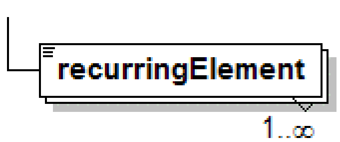
5.3.2 Optional Element

Optional (non mandatory) elements are specified with dashed lines used to frame the rectangle.



5.3.3 Recurring Element

This element (and its child elements if it has any) can occur multiple times.

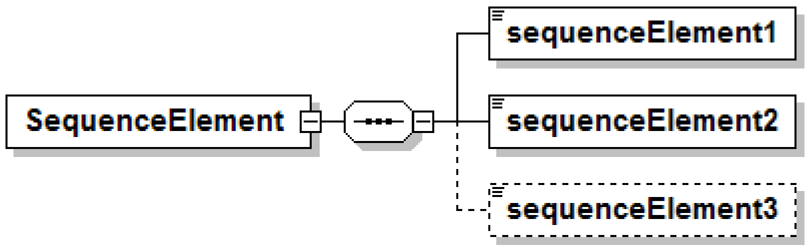


[1] XML Spy: http://www.altova.com

This example shows a recurring element that must occur at least once but can occur an unlimited amount of times. The upper bound here is shown with the infinity symbol.

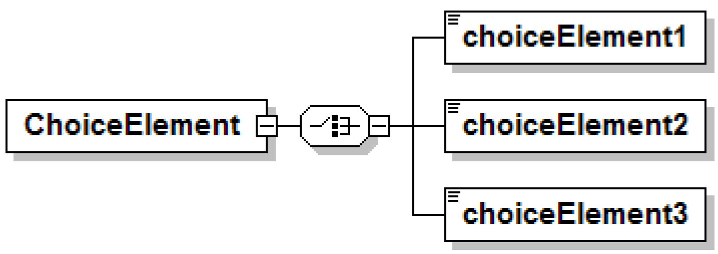
5.3.4 Sequence Connector

The connection box, called a sequence indicator, indicates that the "SequenceElement" data is made up of three elements. In this example, the first two elements are mandatory and the third element is optional



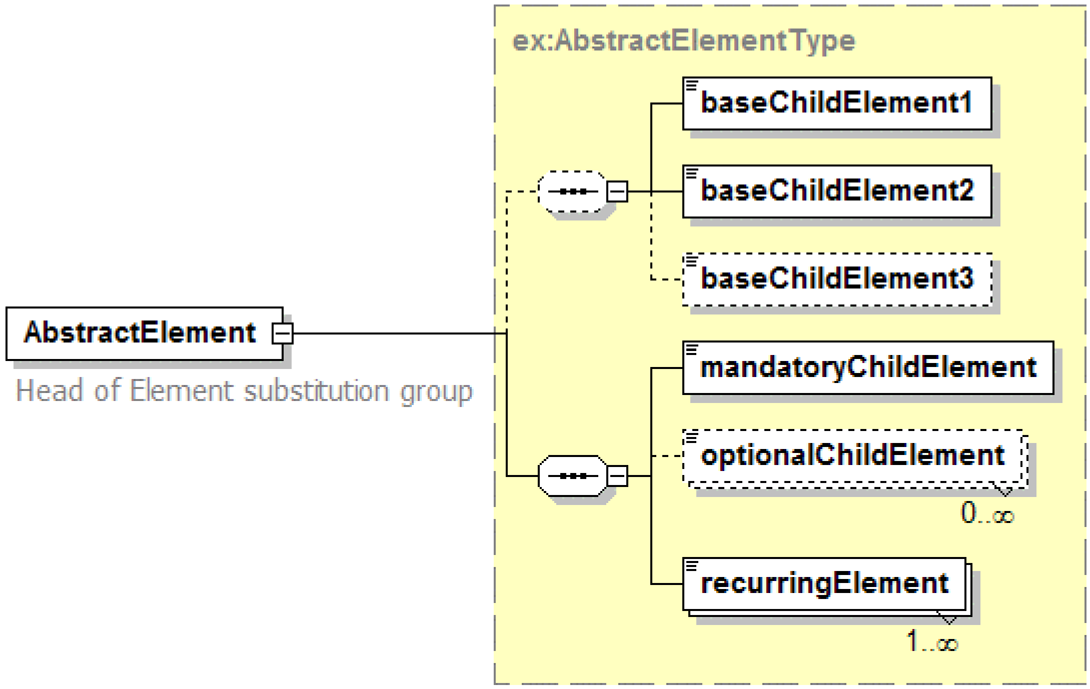
5.3.5 Choice Connector

The connection box here is a "choice" indicator, indicating that there is always going to be exactly one of the child elements listed on the right.



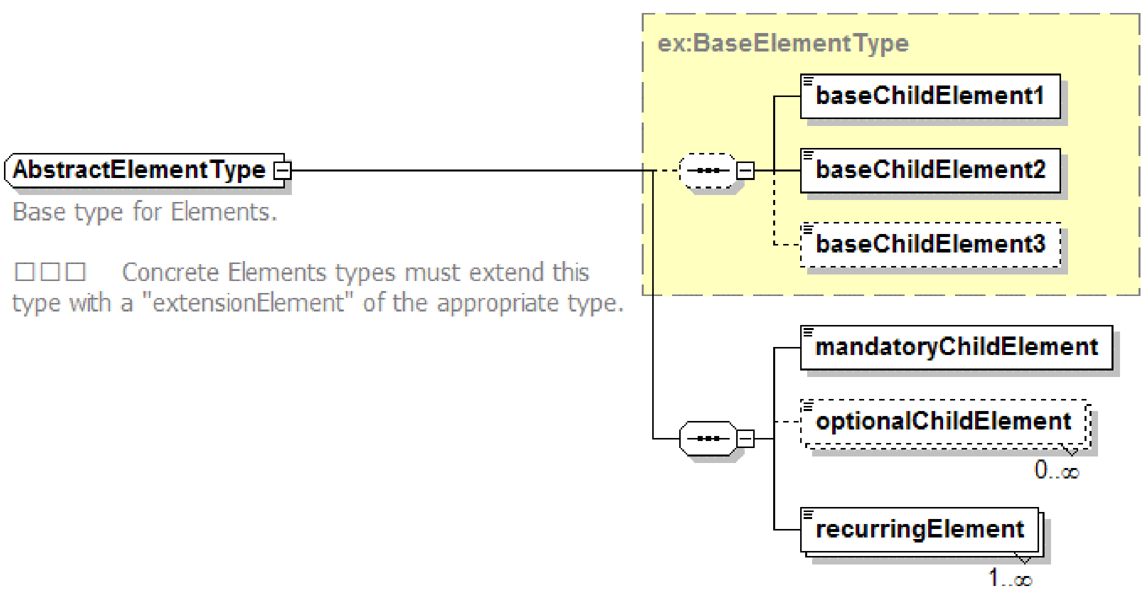
5.3.6 Definition with Complex Type

This diagram illustrates the use of a complex type (i.e., "ex:AbstractElementType") for defining an XML element (e.g., "AbstractElement").



5.3.7 Complex Type

This diagram illustrates the definition of a complex type (i.e., “AbstractElementType”), extending another complex type (i.e.,“ex:BaseElementType”) with three additional elements. Complex types can be reused to specify that different elements are of the same type.



5.4 Used parts of other documents

This document uses significant parts of document [OGC 05-008]. To reduce the need to refer to that document, 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%).

5.5 Platform-neutral and platform-specific specifications

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 specifications. 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 in Annex C, and thus specify the UML model data type and multiplicity of each listed item.

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 GET transfer of operations requests (using KVP encoding), and for use of HTTP POST transfer of operations requests (using XML or KVP encoding). However, the same operation requests and responses (and other data) could be encoded for other specific computing platforms, including SOAP/WSDL.

6 SPS overview

6.1 Introduction

The operational context of the SPS is abstracted from, and therefore applies to, several areas of interest. In the military area there is always a great deal that is unknown about a battlespace, or about a theatre of operations other than war, which gives rise to needs for specific useful information. In the business area corporations and other non-governmental organizations have a need for global economic intelligence. In the scientific area there is a constant interplay between facts, and theories that explain the facts, which then gives rise to the need for more information in order to confirm and extend the theories. Similarly, in the medical area symptoms give rise to a need for information that calls for tests that support diagnosis. All of these areas have information needs, and a common concept of operations can be applied to the satisfaction of those needs. Such operations constitute collection management, that is, management of the process of collecting the needed information.

Effective collection requires a concrete and specific definition of the task or problem and the continuous refinement both of the task and of the information compiled so far, in order to ensure the most comprehensive and accurate collection possible. Such definition and refinement is an essential part of collection management.

Note that parts of the following sub-sections are derived from publicly available information about past military practices (FM34-2, 1994). Those practices were based on a particular philosophy of how to support decision making under uncertainty, and they may or may not be still current. But the practices are still relevant because they have become embodied in various forms in different software systems over the course of time. So they are useful for understanding the functionality provided by those systems, and it is precisely those systems which the SPS is concerned with.

Note also that, even though in the military and business areas it is more usual to talk about intelligence collection than about information collection, this document takes the position that it is really information that is collected, and that intelligence is derived from information in specific user contexts.

6.2 Collection Management

Broadly speaking, collection management (CM) is the utilization and coordination of the resources involved in collecting information. These resources include both operational procedures and systems. One such procedure is the definition and refinement of requirements. One such system is a planning tool. Additionally, collection management may also involve exploitation of raw information, in order to produce information of the quality or specificity required.

The role of the person or agent that performs the activities of collection management is the collection manager (CM). There are three subsets of activities, and corresponding roles that comprise CM. These are requirements management (RM), mission management (MM), and asset management (AM). The SPS standardizes AM activities.

6.2.1 Requirements Management

Requirements management starts with information needs (INs). In the military area, an information need comes from a commander who needs information about a geographic area of interest (AOI). The need for the information can be dependent on what happens when, inside the AOI. In the scientific area, an IN comes from a scientific community, where a theory current in that community gives rise to a question which could be answered by certain information. In the area of medical diagnostics, a patient's case gives rise to a need for information that can help to assign a diagnosis to symptoms.

Requirements management is concerned with turning information needs into information requirements (IRs). Although an information need is associated with a specific reason as to why it is needed, information requirements are even more specific than information needs. An information requirement is precise as to what needs to be known, what is the AOI about which it needs to be known, what are the times about which, and at which, it needs to be known, and by whom it needs to be known.

Requirements management is concerned with the feasibility of information requirements. Are they asking questions that can be answered? If not, it is the task of RM to adjust them or reject them.

Requirements management is also concerned with the process of keeping track of which information requirements are satisfied by which collected information, and which are outstanding. Hence RM is concerned with tracking and expediting the various aspects of collecting information, and with correlating requests with collected information.

6.2.2 Mission Management

Mission management (MM) determines what kind of information can satisfy an IR. MM forms a collection strategy by determining how to satisfy IRs, based on what collection methods are available, and on their suitability to those IRs. MM formalizes the collection strategy into a collection plan. MM derives specific collection requests (CRs) from IRs.

An example from the scientific area illustrates the process of transforming IRs into CRs. In this case, meteorologists had the need to assess the effectiveness of their ability to model both the clear and cloudy sky radiative energy budget in the subtropics and to assess the climate effects of high altitude, optically thin cirrus clouds. This need was translated into the requirement to determine both the radiative properties (such as the spectral and broadband albedos, and infrared emittances) and the radiative budget of cirrus clouds. That was the RM activity. This IR was then split into two different CRs. The Solar Spectral Flux Radiometer was integrated on the NASA Altus UAV and also on the DOE Sandia Twin Otter, in both zenith and nadir viewing modes. That was the MM activity. Both platforms were then tasked to make radiometric observations in parallel. (An AM activity, discussed below.)

6.2.3 Asset Management

Assets management (AM) identifies, uses, and manages available information sources in order to meet information collection goals. AM executes the collection plan by submitting the CRs to the resources involved. This may require resource specific planning. In the medical diagnostics area, the diagnostician fills out a laboratory form which specifies the CRs, and sometimes just sends the patient to the lab with the form. In general, there is a human in the loop, and the SPS will need to be able to take that into consideration.

6.3 Collection Management Process

The collection management process is only one part of the larger collection process, or workflow. The larger process involves:

- Planning and management of the information management and production effort.
- Actual collection and correlation of information.
- Processing of the information, consisting in conversion, rectification, etc.
- Production, or putting the information into a usable from.
- Dissemination of the information to consumers.

Each of the steps in this larger process itself involves complex processing. For example, in the actual collection step, it is customary to identify the following phases.

Phase 1: Collection requests are evaluated and either rejected or else assigned a time window in which they will be executed, so that requestors can make commitments which are contingent on the execution of their request.

Phase 2: sets of collection requests are organized by type and scheduled for execution. This may involve interaction with the requestor.

Phase 3: requests are executed, information is collected and passed through a processing pipeline which ends back at the requestor, or at a point designated by the requestor.

Phase 4: the overall performance of the collection request process is evaluated with respect to its goals (customer satisfaction, or the advancement of science, or so on).

CM per se is just the first step in this larger workflow. It consists in identifying, prioritising, and validating information requirements, translating requirements into observables, preparing collection plans, issuing requests for information collection, production, and dissemination, and continuously monitoring the availability of requested information. In CM, based on the type of information required, on the susceptibility of the targeted activity to various types of collection activity, and on the availability of collection assets, specific collection capabilities are tasked.

7 Concept of Operations

7.1 Existing System Functionality

A number of systems have been developed in support of the various CM roles. The systems themselves tend to come and go over the years, but the basic types of functionality that they provide tend to remain the same. Progress consists in certain capabilities becoming more effective. The following subsections attempt to categorize these capabilities according to the different CM roles, in a summary form.

7.1.1 RM support

- Record, organize, and track collection requirements. Provide feedback on requirements status. Track requirements satisfaction.
- Support aggregating and prioritising requirements.
- Provide feedback for tracking requests, perhaps by querying upstream requirements registries.
- Provide feedback for monitoring the status of production. Provide feedback for monitoring the status of exploitation, if exploitation is part of a requirement.

- Correlate responses with requirements.
- Support determining whether existing data satisfies collection requirements. Support querying databases of previous collection requests and responses.

7.1.2 MM support

- Support developing collection plans. Support allocating collection requirements to collection assets. Coordinate the collection process.
- Provide feasibility models, look-ahead tools, schedule timelines, track and coverage displays.
- Provide platform/sensor models. Provide modelling capability.
- Provide help with evidence-based reasoning.

An interesting example of a support system from the scientific area is the Scientist's Expert Assistant (SEA) system from NASA. This system assists scientists in formulating CRs targeted at the Hubble space telescope. Among other things, it allows an investigator to delineate a portion of the sky to be observed, choose bright radiation sources that need to be excluded, calculate the tiles of a mosaic that covers the delineated segment, and assign them to orbits, and to simulate the properties of filters for the selected observations.

7.1.3 AM support

- Perform actually tasking. This may involve generating and dispatching tasking and request messages, and providing assistance in preparing such messages, something that can range anywhere from basic message syntax checking to more advanced feasibility checking, which determines in advance if it is going to be possible for an asset to handle a request.

7.2 Making Existing Functionality Interoperable

The goal of the SPS is to make the types of functionality that exist to support the AM role interoperable. This has several implications, the first of which is that CM systems will only have to interact with one kind of interface. It also means that CMs will be able to use any support system that has been "wrapped" with the SPS interface, including new systems. For example, critical information, such a meteorological information, that has not traditionally been considered part of intelligence, can be integrated into the CM process in this way. Finally, it means that different kinds of SPS client applications can be used with the same SPS services. For example, one kind of client might be a browser-based Web client, while another kind of client might be a workflow system that manages the CM roles, their interactions, and their products. For any given SPS, these clients will be interchangeable.

7.3 Interacting Workflows

As mentioned above, the CM process can be thought of as a workflow, and might even be implemented with a workflow management system. On the other hand, each of the assets that the CM process interacts with is also likely to involve another complex workflow. Consider the case of a sensor on a UAV.

The process of getting an observation from the sensor at the right time and place involves many other activities which the CM never sees. These include flight planning, flight plan approval processes and airspace coordination, payload planning, air traffic control, and frequency planning for both vehicle and sensor control and for data transmission. There are a number of roles here, including the UAV operator, the payload operator, the UAV mission planner, the traffic controllers, and others, which are part of another workflow.

This other workflow is the asset management system, and the fundamental unit of work in this workflow is the flight. The activities of this workflow have the common goal of making the flight successful, where what normally counts as a successful flight is one in which the UAV and the sensors that it carries have been maximally utilized. Maximal utilization is a goal that involves allowing the greatest number of compatible collection requests to the flight. This means that some collection requests should be cancelled or delayed, rather than allowing them to subtract from the total utilization provided by a better set of different requests.

Clearly these two interacting workflows have different fundamental units of work, and different and conflicting goals. SPS serves as a bridge between these different units, and provides a way to balance between the different goals. One way in which the SPS bridges the gap is through the exchange of information about feasibility. (See below.)

In summary, SPS is the interface between the CM process (or workflow) and the – in this case – UAV process (or workflow). The goal of the CM process is to satisfy information needs. It does this by asking other workflows, such as the UAV process just described, to collect, to process, and to deliver information. This other workflow is not the asset, it is how the asset is used. In what follows this other workflow will be called the asset support system. The SPS provides a standard interface to different kinds of asset support systems, from different kinds of CM processes (or workflows). In what follows, the CM process will be referred to as the SPS client, or sometimes more informally as the “user”.

7.4 Simulation

One important category of asset is simulation. Typically simulation management works differently than other AMS's do, and the SPS needs to be flexible enough to also handle these.

7.4.1 Basic concepts

The term *simulation* is defined as the use of models to investigate time dependent processes. Following this definition, real or fictive systems will be described in the form of conceptual models. Based on analysis and abstraction, these models should describe the circumstances in a formal and unambiguous way (Zeigler et al., 2000).

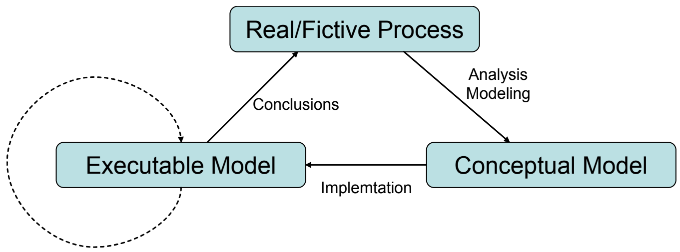


Figure 1: Models and Simulation, according to [9], modified

The implementation of a conceptual model by an executable model generates a way to execute experiments[2]. The results of these experiments allow conclusions about the modelled system. In the case that the executable model is made up of software (software model), it is called a *computer simulation*[3]. The following comments apply to computer simulations exclusively. A program that executes a simulation model is called a *simulator*.

Regarding simulation in general, there exist numerous approaches which focus on different areas of usage. Zeigler et al. describe an abstract framework for modelling and simulation (Zeigler et al., 2000). This framework is build up from definitions of fundamental entities and their relationships to one another. The foundation consists of the real or virtual (source) system that we are interested in modelling. It is viewed as a source of observable data, in the form of time-indexed trajectories of variables. The variables are stored into a *system behavior database*. The specifications of the conditions under which the system is observed or experimented with build the *experimental frame*.

Zeigler et al. differentiate between the entities model and simulator (Zeigler et al., 2000). Rather generally speaking, a model will be defined as a set of instructions, rules, equations, and constraints that will consequently build up an I/O-behavior consistent with the source system. A simulator is considered to be an agent capable of actually obeying the instructions, generating the model's internal behavior and eventually executing the model.

In practice, it is often only required that the model faithfully capture the system behavior to the extent demanded by the objectives of the simulation study, within an acceptable tolerance, as it is mostly impossible to achieve fully validity of the model (Kleindorfer et

[2] A (simulation) experiment may consist of any number of individual simulation runs. Those individual simulation runs allow an estimation of the simulation model response to specific parameterisation.

[3] An alternative to computer simulations would be simulations which are based on physical models (e.g. scale- or imitation models).

al., 1998; Oreskes et al., 1994). Furthermore, the *simulator correctness* demands that the simulator executes the model correctly. Correctness is fulfilled if it is guaranteed that a simulator faithfully generates the model's output trajectory given its initial state and input trajectory.

If concept or validity and simulator correctness are fulfilled for a specific experimental frame, the real system and the simulator could be viewed as interchangeable.

Simulation methods are traditionally classified regarding the way in which the simulated time progresses and the way that the status of the simulated system is described and modified.

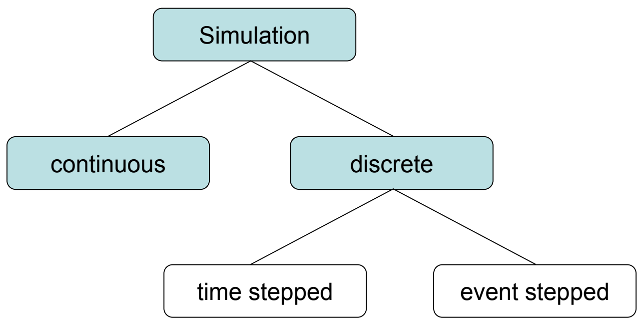


Figure 2: Taxonomy of conventional simulation methods, according to [10].

Roughly, simulation models can be categorized into discrete and continuous approaches. (Time-) discrete approaches are characterized by the fact that the simulation time advances in jumps from one point in time to the next. The model status changes atomically. On the contrary, the simulation time within continuous simulation models changes gradually, means that status variables of the model change within a finite period of time infinite times. As a rule continuous simulation models are based on partial differential equations, with the free variable being the time (Klein).

Regarding a simulation, Fujimoto distinguishes three different fundamental terms of *time* (Fujimoto, 1999):

1. *physical time*, that describes the real time, applied to the modelled system

2. *simulation time*, which is generated as a "virtual time" within a simulation model. It is an abstraction of the physical time. It is defined as a complete, ordered quantity wherein each value defines a moment in time within the modelled system. For all simulated points in time t_1 and t_2 that describe the physical points in time p_1 and p_2 is true that, if $t_1 < t_2$, hence p_1 occur before p_2 and $(t_2\text{-}t_1) = (p_2 - p_1) * k$, with $k = constant$. The linear relation between the simulation time and the physical time provides a faithful correspondence between simulated periods of time and its real counterparts.

3. *wallclock time* describes the real time during a simulation run.

7.4.2 Integration into Web Services

Becuase of the fact that a simulator does not differ from a sensor with respect to the provision of spatio-temporal data (it only differs in the way in which it estimates the requested value and its virtually temporal independence), it is allowed to use simulators instead of sensors. Two different scenarios can be distinguished. In the first case, the simulator is continuously running, independently of the user. It is started, parametrized and maintained by the data provider and has no interface to the user other than at the point of data access. These simulators are e.g. a weather forcast that provide values for the parameters temperature, precipitation and wind speed, or air or water quality assessments. It is even possible that a user will not even notice that the provided data results from calculations rather than from real measurements (e.g. if the values are interpolated from a wide-ranging measuring network). Those simulators will be encapsulated by a simple datastore that is accessible using a Sensor Collection Service. It can be handled purely synchronously and follows the service trading (publish – subscribe – bind) paradigm.

In the second case, a simple encapsulation is insufficient. All observations or even simulations that require preceding feasibility studies, complex control and management activities, or intermediate and/or subsequent user notifications, can not be handled synchronously anymore, but become heavily asynchronous. In this second case the service interactions become much more complex. A list of services will become necessary. The SPS handles the main part of the simulation. The SPS provides interfaces that allow requesting a simulation run, feeding the necessary parameters to the simulator and to start the simulation run.

Sensor Planning Services as facades for simulation models are the first step within the integration process of simulators into web services. Currently, a rather tight coupling is unavoidable. This situation will be improved when the first generic simulation interfaces become available, providing simulation management that is independent of the underlying simulation system.

8 Transactions

The Sensor Planning Service interface facades often complex asset management systems that do not provide an immediate response to GetFeasibility request operations as the asset has to be analysed first. This might be a time consuming task. In other cases wants a client to task an asset and wants to be informed in the moment the data is available for retrieval. Either case, we have to deal with long term transactions that require the implementation of an asynchronous interaction pattern.

Assuming that the application level of the OSI protocol stack is where HTTP resides, then there are conceptually three additional protocol layers (or perhaps they should be called application level sublayers) in the SPS transaction model.

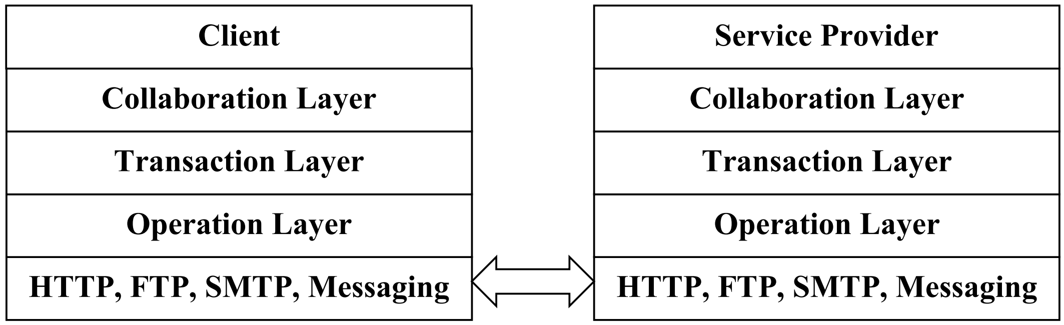


Figure 3: Protocol layers

The operation layer uses the basic application layer protocol – HTTP, for example – to implement the request /response pairs that form the OGC operations such as are used in the GetMap or GetFeature interfaces of WMS or WFS. The transaction layer uses the operations in the operation layer to implement long-term or "advanced" transactions. At the collaboration layer transactions are "choreographed" into collaborations.

An example of a transaction is the interaction which takes place when a request for information is submitted through a SPS, and is retrieved ("committed") through some other service; then a third service provides the notification that the results are ready.

An example of a collaboration is the interaction which takes place between a client, a registry, and a service during the "publish, find, and bind sequence". Another example of a collaboration is the interaction between a transaction which returns a schema, and the transaction that uses that schema to formulate a query. (In both of these cases it is principally the client software that maintains the state of the collaboration.) For the SPS, there is an important collaboration between the DescribeTaskingRequest operation and the Submit transaction.

8.1 Short Term Transactions

Short term transactions consist of a single operation. They can be thought of as atomic transactions that either successfully return a result (analogous to a database „commit“ or return an exception report (analogous to a "rollback").

8.2 Long Term Transactions

Long-term transactions, which are sometimes called "long-running conversations", consist of more than one operation, typically involving more than one OGC interface. In the case of SPS, instances of such transactions may take days to complete, and require that it be possible to track their progress, and if necessary, to cancel them.

The following diagram shows the interactions which take place in an asynchronous collection request to an SPS to get observation data. Participants in grey represent non-OCG services or processes, such as legacy, or proprietary, asset support systems for

planning, scheduling, tasking, collecting, processing, archiving, and delivery of information requests and usable information, or (in the case of feasibility determination) for assisting in the use of such systems. Other participants are an OGC enabled client, called here “User”, and three services, the SPS, an SOS (which provides standard access to the data once it has been collected), and a notification service (WNS) which notifies the User when the SOS is ready with the requested data.
The diagram shows an initial stage where the user determines the capabilities of the SPS (in this case, without using a Registry). Then the user registers with a WNS so that the SPS understands the user’s preferences for notification delivery. It then shows the user submitting the actual request. The requested task will be started. On condition that everything runs smoothly, the asset will start its observation and stores its observational data in a datastore. If the task is finally completed, the asset management system (AMS) will send an ASM-specific message to the SPS. As a result, the SPS will send a doNotification request to the WNS where the user is registered to inform him that (and where) is ready for retrieval. It is up to the client to send the getObservation request to the SOS in order to access the data.

There are several important pieces of information that move through the transaction. These are:

WNSData. This associates to a user the following information: the URL of the WNS where the user is registered and the WNSRegistrationID that was provided by that WNS. It is the means through which notifications are sent to the user. Therefore it has to be known to the SPS so that the same user who submitted a request can be notified when that request completes (or any kind of failure occurs which is not shown here).

TaskID. This identifies a request to its results through the transaction. It is either established by SPS and sent to AMS (at step 10), or established by AMS and sent to SPS. It is held by the user, to be used to check status, update or cancel a request (not shown here).

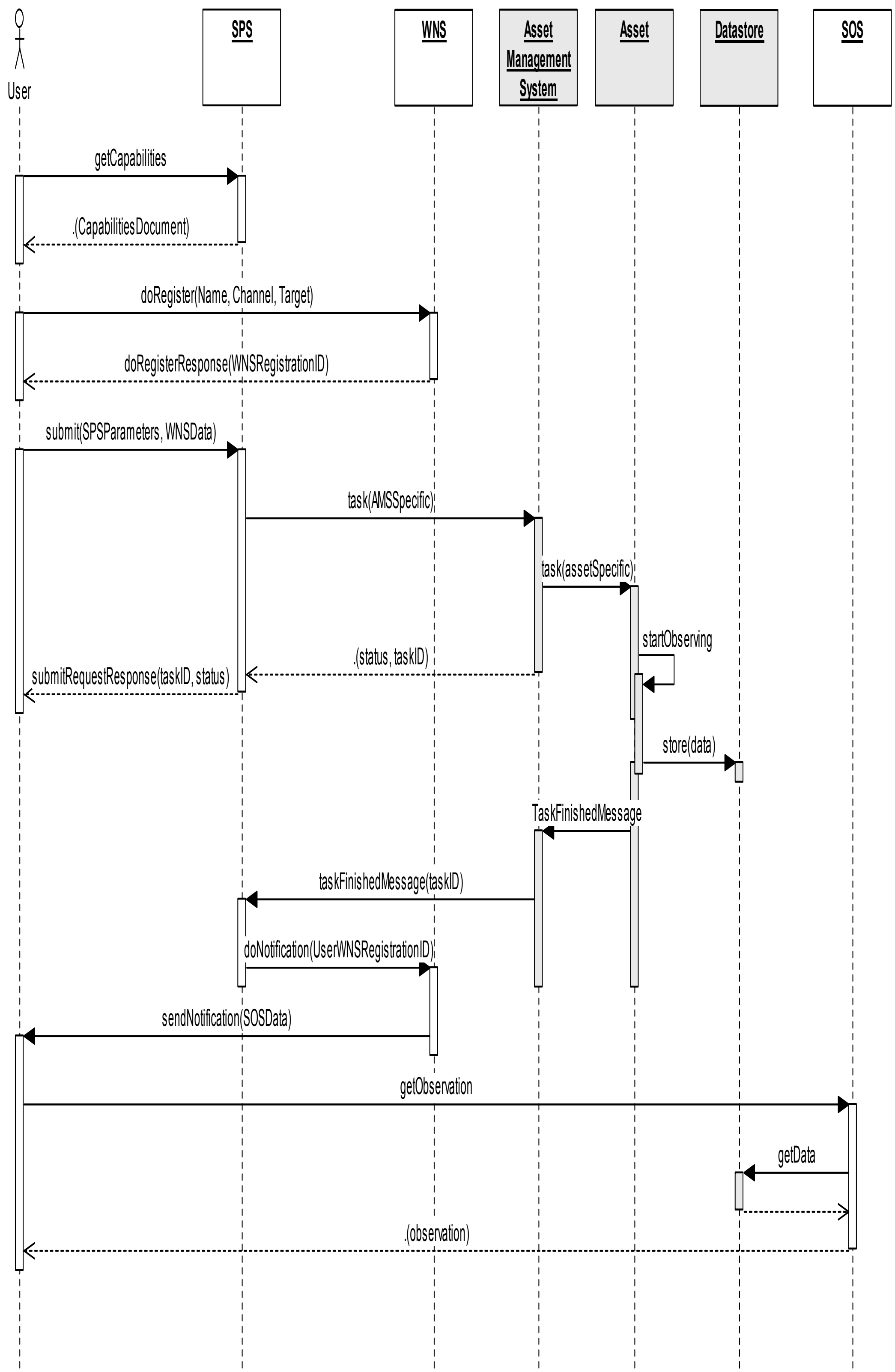

User
SPS
WNS
Asset Management System
Asset
Datastore
SOS
getCapabilities
.(CapabilitiesDocument)
doRegister(Name, Channel, Target)
doRegisterResponse(WNSRegistrationID)
submit(SPSParameters, WNSData)
task(AMSSpecific)
task(assetSpecific)
startObserving
.(status, taskID)
submitRequestResponse(taskID, status)
store(data)
TaskFinishedMessage
taskFinishedMessage(taskID)
doNotification(UserWNSRegistrationID)
sendNotification(SOSData)
getObservation
getData
.(observation)

Figure 4: Simplified sequence diagram in UML notation showing a long term transactions using additional OGC Web services (non OGC specified elements in grey)

Datastore and **SOS_URL**. Either the SPS knows these and tells AMS, or else SPS knows them and knows that AMS knows them. In the future the user may be allowed to establish them.

9 SPS Parameters: Internal and external representation

One of the core functionalities provided by SPS is the unambiguous presentation of the parameters that have to be set in order to task what ever system is façaded by SPS. The aspects meaning and encoding of those parameters play a crucial role as it comes to interoperability between SPS server and client. To support any kind of asset system, independently of its nature as a physical sensor, an actuator or a simulation model, the parameter description and encoding requires much flexibility.

It is a common design goal of the SWE framework to utilize common data structure and encodings to achieve a higher level of compatibility and reuse of software between the various encodings and services. The Sensor Web Enablement Common namespace (swe) includes definitions that are expected to be shared among all SWE encodings and services. It defines several basic value types and data encoding as well as aggregate data types that group any simple basic types. Both simple and aggregate types form the basis for the SWE Common *DataDefinition,* a schema to explicitly describe the data components expected and the encoding of values. For further information on the different data types as well as on *DataDefinition* see document OGC 05-086, Sensor Model Language.

The use of *DataDefinition* elements is one option to define content and encoding of parameters. Another option that may serve the needs of specific information communities in a more effective way is the use of task messages that are organized in remotely stored message dictionaries and consist of any number of well defined elements. Those remote schemas will define meaning and parameter payload for specific parameters. We will illustrate this option considering a dictionary we will call Aircraft Control Tasking Messages (ACTM) as an example. The ACTM basically consist of a number of simple and complex element definitions that are stored in a registry. The registry might be organised as a simple schema file stored on a web server or as part of a more complex system. The following listing shows an extract of the ACTM schema.

```
<xs:simpleType name="TargetElevation">
    </xs:annotation>
    <xs:restriction base="xs:decimal">
        <xs:minInclusive value="-500"/>
        <xs:maxInclusive value="10000"/>
        <xs:pattern value="^[+-][0-9](\.[0-9]+){5}$"/>
    </xs:restriction>
</xs:simpleType>
```

Listing 9-1: Part of the ACTM dictionary

We define a simple type with name *TargetElevation*. A SPS facading an asset that requires a value for this element will use a reference to the remotely stored ACTM schema to indicate that a parameter of type *TargetElevation* is needed. The specific element will be identified by its name attribute. For this reason, the name attribute has to be used unambiguously throughout the remote schema. The advantage of remotely defined schemas is that the possible values used in instance documents can be further restricted to match specific needs of user communities. In our example, the instant document has to match the following criteria:

1. The value has to be of type xs:decimal
2. The minimum value is set to -500, the maximum value to 10000
3. The value has to be encoded following a specific pattern: one + or – symbol followed by maximal positive numbers in the range from 0 to 9, separated by "." symbols.

The simple types can be agglomerated to more complex types. Figure 5 illustrates a more complete excerpt of the ACTM schema. We see that a SPS may even require an instance of an ACTM CriticalMission message in order to task its asset. In contrast to a non-critical message, the critical message requires a more complex MissionID pattern, e.g. 01-102, whereas the MissionID of a non-critical message is just two digit integer.

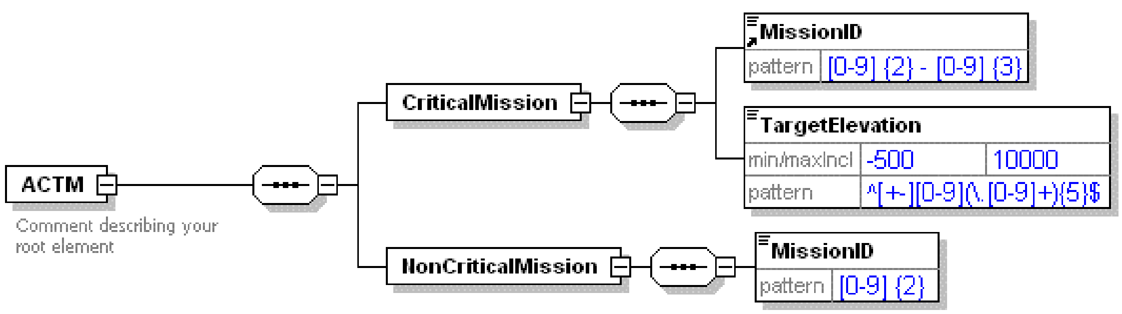


Figure 5: ACTM schema in XMLSpy representation

A SPS server may define an urn:community:dict:CriticalMission element as a mandatory parameter. It can be assumed that those dictionaries are usually used by clients and servers that "know" how to encode the elements rather than accessing the schema definition every time in order to get information about proper encoding. Though, this is still an option.

10 SPS Operations

10.1 SPS Operation Overview

The SPS operations can be divided into informational and functional operations. The informational operations are the GetCapabilities operation, the DescribeTasking operation, the DescribeResultAccess operation, and the GetStatus operation. Among these, the GetCapabilities, the DescribeResultAccess and the GetStatus operations provide information that the SPS user needs to know, while the DescribeTasking operation provides a description of information that an asset management system needs to know. The functional operations are the GetFeasibility operation and the Submit, Update, and Cancel operations. All of these operations have an effect on the asset management system, as explained below.

The SPS interface (currently) specifies eight operations that can be requested by a client and performed by a SPS server. Those operations are:

a) GetCapabilities (mandatory) – This operation allows a client to request and receive service metadata (or Capabilities) documents that describe the abilities of the specific server implementation. This operation also supports negotiation of the specification version being used for client-server interactions.

b) DescribeTasking (mandatory) – This operation allows a client to request the information that is needed in order to prepare an assignment request targeted at the assets that are supported by the SPS and that are selected by the client. The server will return information about all parameters that have to be set by the client in order to perform a Submit operation.

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

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

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

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

g) Cancel (optional) – This operation allows a client to cancel a previously submitted task.

h) DescribeResultAccess (mandatory) – This operation allows a client to retrieve information how and where data that was produced by the asset can be accessed. The server response may contain links to any kind of data accessing OGC Web services such as SOS, WMS, GVS, or WFS.

These operations have many similarities to other OGC Web Services, including the WMS, WFS, and WCS. Many of these interface aspects that are common with other

OWSs are thus specified in the OpenGIS® Web Services Common Implementation Specification [OGC 05-008]. Many of these common aspects are normatively referenced herein, instead of being repeated in this specification.

Figure 1 is a simple UML diagram summarizing the SPS interface. This class diagram shows that the SPS interface class inherits the getCapabilities operation from the OGCWebService interface class, and adds the SPS operations. (This capitalization of names uses the OGC/ISO profile of UML.) A more complete UML model of the SPS interface is provided in the following subclauses.

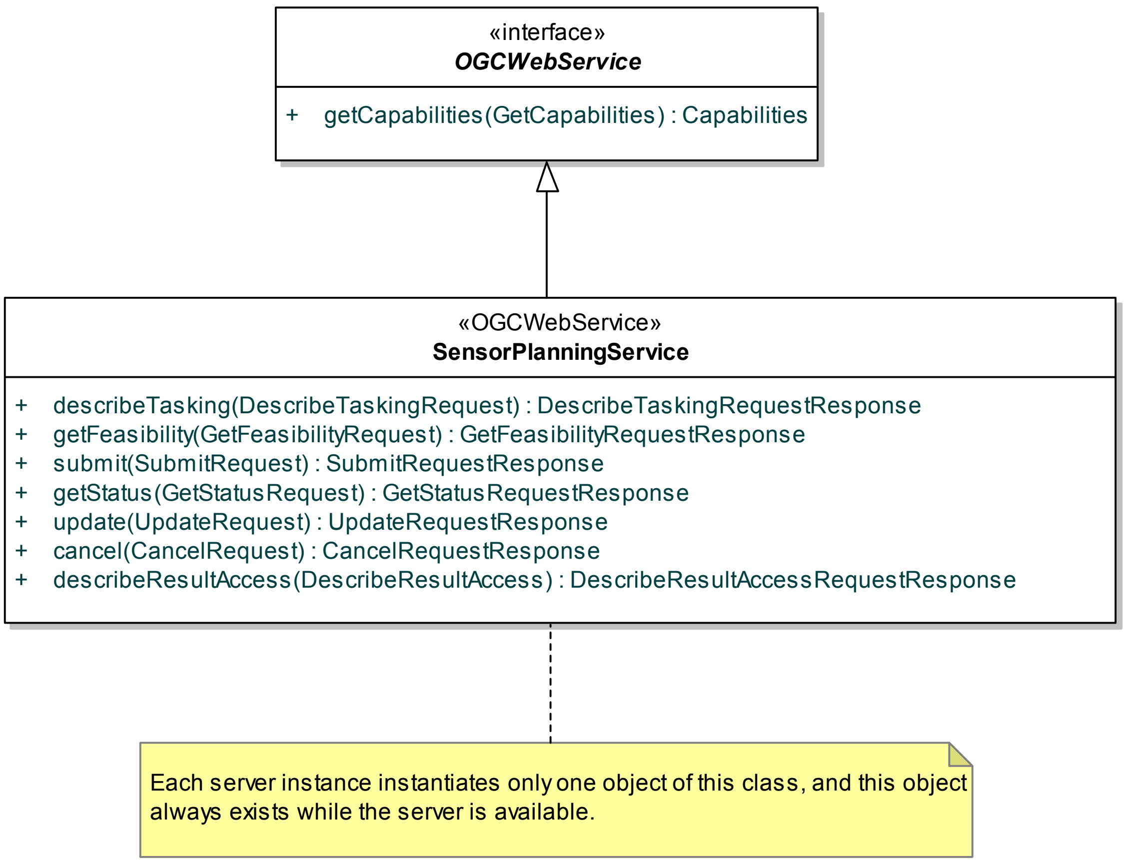


Figure 6: SPS interface UML diagram

NOTE In this UML diagram, the request and response for each operation is shown as a single parameter that is a data structure containing multiple lower-level parameters, which are discussed in subsequent clauses. The UML classes modelling these data structures are included in the complete UML model in Annex C.

10.2 SPS Operations Usage

Each of the eight SPS operations is described in more detail in subsequent clauses. To simplify the understanding of the information model behind the different operations, some annotated sequence diagrams in UML notation will follow.

The first diagram illustrates the first steps that usually occur during a SPS interaction. An actor (which might be a user with a client GUI or another service) first requests an overview of the capabilities of the SPS. It provides information about the phenomenons and sensors that can be tasked using this SPS. If the user wants further information upon access of the data that is produced by the façaded sensors, a DescribeResultAccess operation would be used. In this example, we presume that the user received sufficient information as part of the Capabilities document at this stage. Therefore, the next step is to find out which parameters have to be provided to task an asset. This is done using the DescribeTasking operation. The response provided by the SPS defines the parameters that have to be set for each sensor.

Next might be a GetFeasibility request to obtain information if a specific task is feasible. This operation is primarily used to find out if a specific task is likely to be executed under the current conditions. Imagine a UAV that shall make some pictures of a specific area. It has to be checked if the requestor is allowed to operate the UAV, if the requested area fits into the UAV flight plan or if it could be adapted to fit etc. If the GetFeasibilityRequestResponse indicates that the request is likely to be executed (there might be always some late change in the conditions that enforce a previously feasible task not to be feasible anymore; e.g. if another requestor with a higher priority wants to fly the UAV into the opposite direction), the actor will send the Submit request. The Submit request may contain the feasibilityID or data for all parameters.

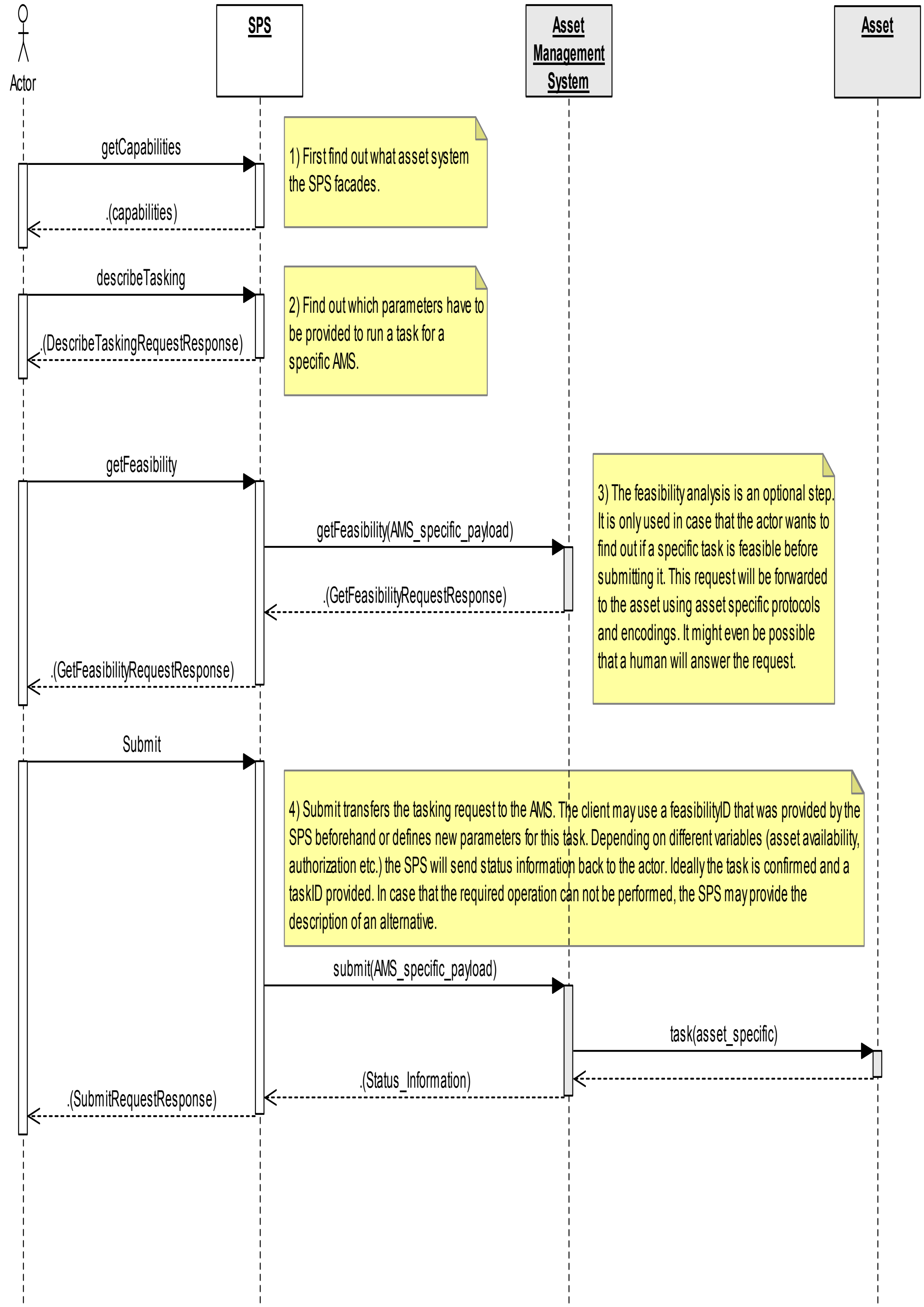

Actor
SPS
Asset Management System
Asset
getCapabilities
.(capabilities)
1) First find out what asset system the SPS facades.
describeTasking
.(DescribeTaskingRequestResponse)
2) Find out which parameters have to be provided to run a task for a specific AMS.
getFeasibility
getFeasibility(AMS_specific_payload)
.(GetFeasibilityRequestResponse)
.(GetFeasibilityRequestResponse)
3) The feasibility analysis is an optional step. It is only used in case that the actor wants to find out if a specific task is feasible before submitting it. This request will be forwarded to the asset using asset specific protocols and encodings. It might even be possible that a human will answer the request.
Submit
4) Submit transfers the tasking request to the AMS. The client may use a feasibilityID that was provided by the SPS beforehand or defines new parameters for this task. Depending on different variables (asset availability, authorization etc.) the SPS will send status information back to the actor. Ideally the task is confirmed and a taskID provided. In case that the required operation can not be performed, the SPS may provide the description of an alternative.
submit(AMS_specific_payload)
task(asset_specific)
.(Status_Information)
.(SubmitRequestResponse)

Figure 7: Annotated sequence of the usual steps occurring to submit a task (UML notation)

The next diagram shows how to interact with a SPS after a task was submitted successfully. We see the update and the describeResultAccess operation that was introduced with this version of the SPS specification.

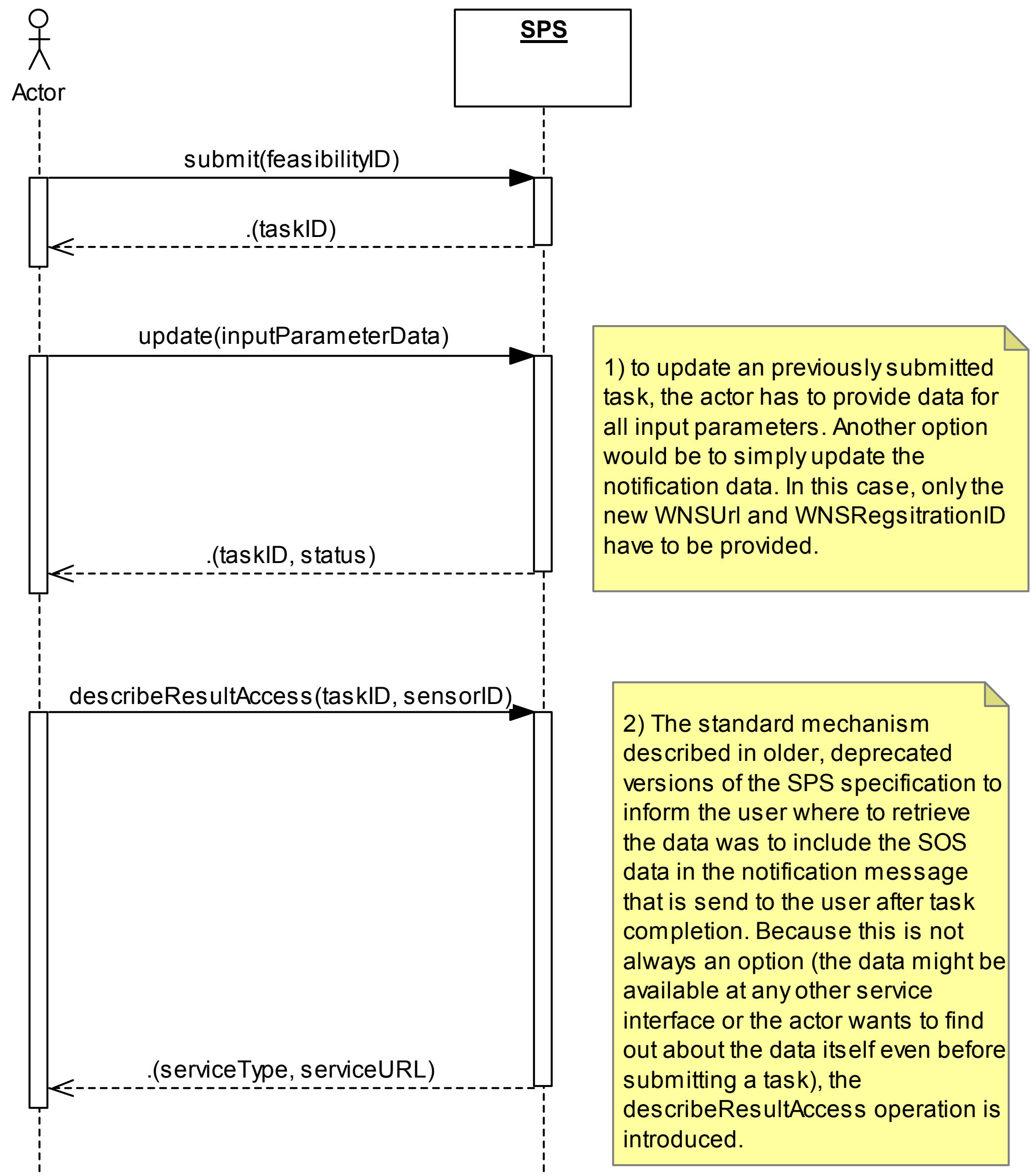


Figure 8: Annotated sequence showing the update and describeResultAccess operations in UML notation

11 Shared aspects

11.1 Introduction

This clause specifies aspects of the SPS Service behavior that are shared by several operations.

11.2 Shared operation parameters

This clause specifies some of the parameters used by multiple operations specified in the following clauses. The parameter names, meanings, data types, and multiplicity shall be as specified in Table 1.

Table 1 — Definitions of some operation request and response parameters

Name	Definition	Data type and value	Multiplicity and use
InputDescriptor	Defines the input required to task a sensor	complex type	-
notificationTarget	Defines the WNS that has to be used to notify the client about the request results	complex type	-

11.2.1 InputDescriptor

The InputDescriptor defines the input a client has to provide to task an asset. The following UML model and the following figure provide an overview of the InputDescriptor.

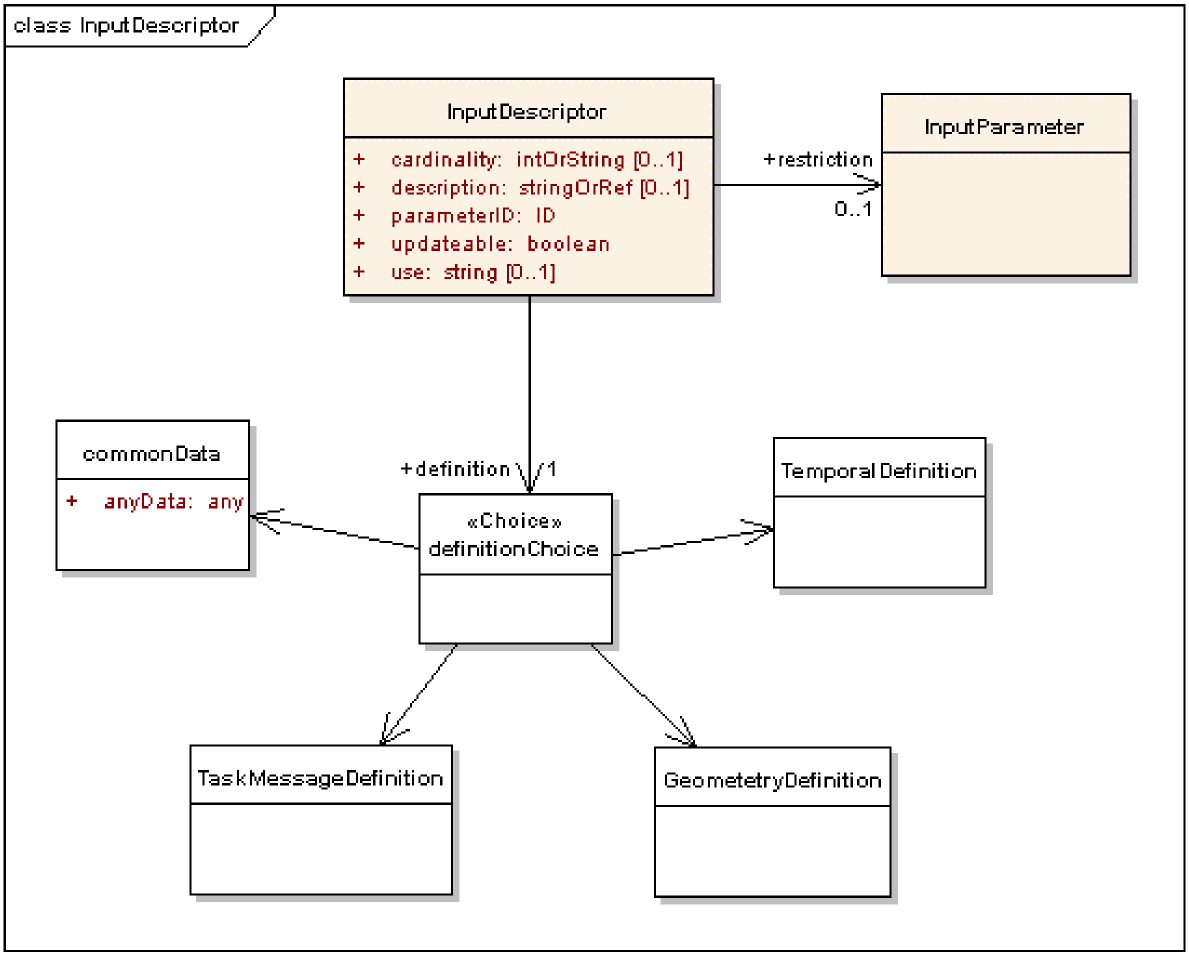


Figure 9: InputDescriptor in UML notation

<u>InputDescriptor: Attributes</u>

An InputDescriptor contains the mandatory attributes "parameterID" which will be used to reference a specific parameter in other requests, e.g. Submit. The "use" attribute defines if the parameter must or shall be provided by the client. It enumerates the strings "optional" and "required". If required, a Submit, Update or GetFeasibility request will not be validated as true if this parameter is missing. The third attribute "updateable" is optional and defines if a parameter can be updated by subsequent UpdateRequest requests.

<u>InputDescriptor: Elements</u>

The InputDescriptor defines four elements: Three optional elements and one mandatory. The three optional elements can be used to provide further description about this InputDescriptor (**gml:description**), allow the presetting of possible values (**restriction**) (e.g. "yes, no" or "day, night" or "1, 2, 3"), and define the cardinality of possible input elements (**cardinality**). Cardinality is restricted to positive integers (excluding zero) and the string value "unbounded".

The data structure of the input elements that shall be provided by the client is defined in the mandatory "definition" element. This element serves as an entry point to parsers to find the data block definition that has to be matched by the input data. It is followed by either a swe:DataDefinition element (see SWECommon for further information), a "taskMessageDefinition" element which is a link to an external definition of the data block or a GeometryDefinition or TemporalDefinition. The latter two are of type QName and are restricted to the GML elements gml:Point, gml:Line, gml:Polygon or gml:TimeInstant and gml:TimePeriod respectively. It is assumed that clients "know" how to encode those basic elements.

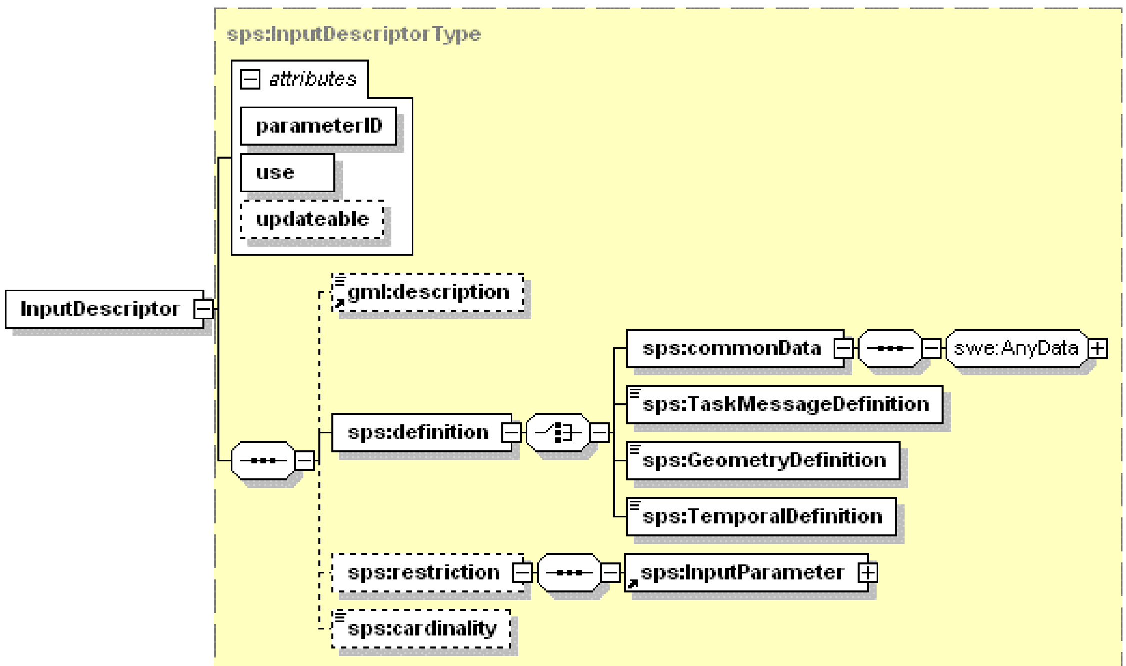


Figure 10: InputDescriptor Element in XMLSpy notation

11.2.2 InputParameter

The InputParameter Element is used to provide the value for a specific parameter. The encoding follows the description that is part of the definition element of an InputDescriptor Element (see subclause 11.2.1). The InputParameter Element is therefore rather simple in its definition. It just has to provide the mandatory parameterID attribute to link the values to the specific parameter. The values itself are replacing the any-Element.

The following figures illustrate the InputParameter element in UML and XMLSpy notation.

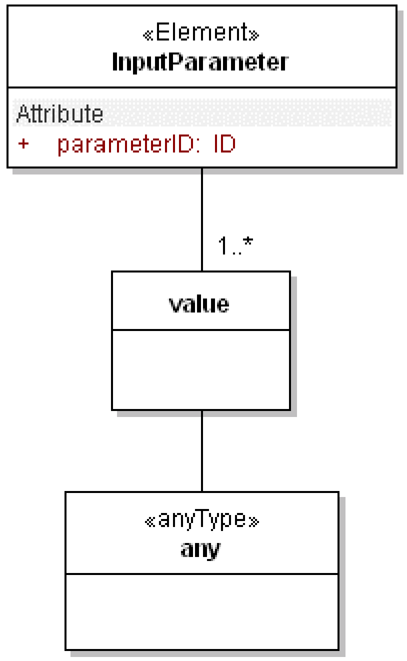


Figure 11: InputParameter in UML notation

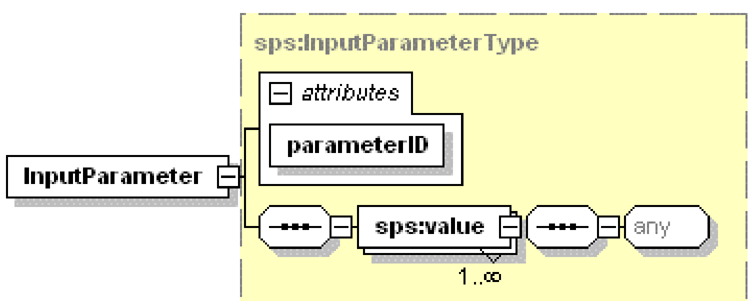


Figure 12: InputParameter in XMLSpy notation

11.2.3 NotificationTarget

The notificationTarget parameter is used to identify the Web Notification Service that has to be used to send information to the client. Using the WNS, notifications may be sent asynchronously to the client.

The following figures illustrate the notificationTarget parameter.

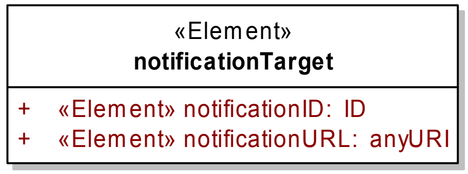


Figure 13: notificationTarget parameter in UML notation

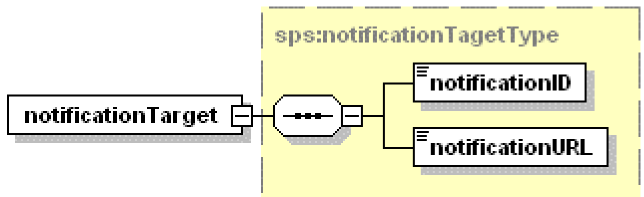


Figure 14: notificationTarget parameter in XMLSpy notation

11.3 Operation request encoding

The encoding of operation requests shall use HTTP GET with KVP encoding and HTTP POST with XML and/or KVP encoding as specified in Clause 11 of [OGC 05-008]. Table 2 summarizes the SPS Service operations and their encoding methods defined in this specification.

Table 2 — Operation request encoding

Operation name	Request encoding
GetCapabilities	KVP and optional XML
DescribeTasking	XML
GetFeasibility	XML
Submit	XML
GetStatus	XML
UpdateRequest	XML
Cancel	XML
DescribeResultAccess	XML

12 GetCapabilities operation (mandatory)

12.1 Introduction

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

12.2 Operation request

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

The “Multiplicity and use” column in Table 1 of [OGC 05-008] specifies the optionality of each listed parameter in the GetCapabilities operation request. Table 3 specifies the implementation of those parameters by SPS clients and servers.

Table 3 — Implementation of parameters in GetCapabilities operation request

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

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

EXAMPLE 1 To request a SPS capabilities document, a client could issue the following KVP encoded GetCapabilities operation request with near-minimum contents:

http://mars.uni-muenster.de/SPS/SPS?Request=GetCapabilities&Service=SPS

EXAMPLE 2 The corresponding GetCapabilities operation request XML encoded for HTTP POST is:

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

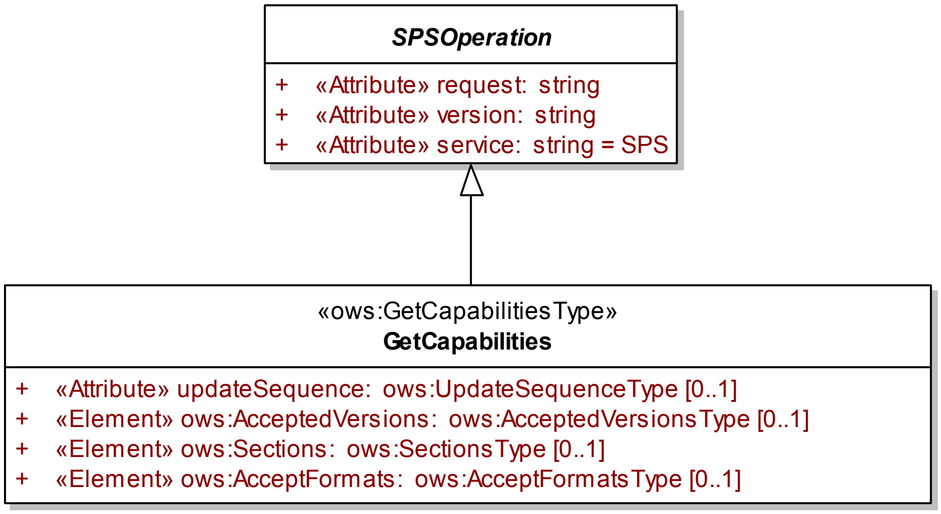



Figure 15: GetCapabilities request in UML notation (normative)

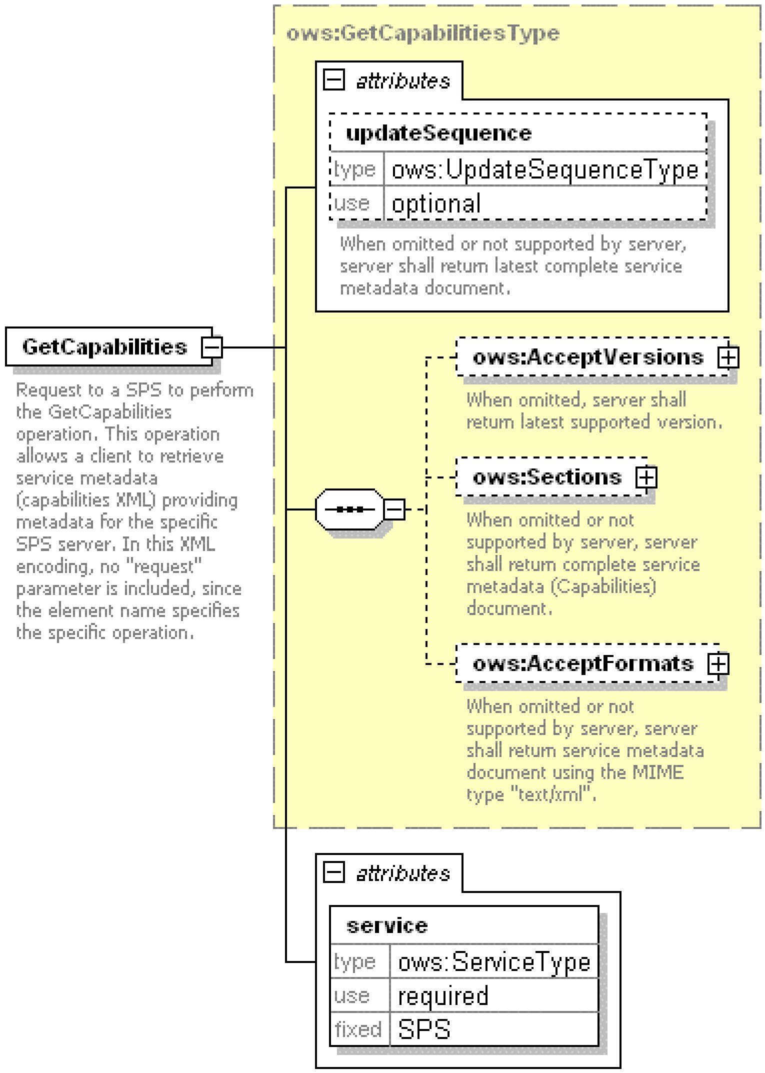


Figure 16: GetCapabilitiesRequest in XMLSpy notation (informative)

12.3 GetCapabilities operation response

12.3.1 Normal response

The service metadata document shall contain the SPS sections specified in Table 4. Depending on the values in the Sections parameter of the GetCapabilities operation request, any combination of these sections can be requested and shall be returned when requested.

Table 4 — Section name values and contents

Section name	Contents
ServiceIdentification	Metadata about this specific server. The schema of this section shall be the same as for all OWSs, as specified in Subclause 7.4.3 and

	owsServiceIdentification.xsd of [OGC 05-008].
ServiceProvider	Metadata about the organization operating this server. The schema of this section shall be the same for all OWSs, as specified in Subclause 7.4.4 and owsServiceProvider.xsd of [OGC 05-008].
OperationsMetadata	Metadata about the operations specified by this service and implemented by this server, including the URLs for operation requests. The basic contents and organization of this section shall be the same as for all OWSs, as specified in Subclause 7.4.5 and owsOperationsMetadata.xsd of [OGC 05-008].
Contents	Metadata about the data served by this server. For the SPS, this section shall contain information about the sensors that can be tasked and the phenomena that can be measured by these sensors, as specified in Subclause 12.3.3 below.

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

12.3.2 OperationsMetadata section standard contents

For the SPS, the OperationsMetadata section shall be the same as for all OGC Web Services, as specified in Subclause 7.4.5 and owsOperationsMetadata.xsd of [OGC 05-008]. The mandatory values of various (XML) attributes shall be as specified in Table 5. Similarly, the optional attribute values listed in Table 6 shall be included or not depending on whether that operation is implemented by that server. In Table 5 and Table 6, the "Attribute name" column uses dot-separator notation to identify parts of a parent item. The "Attribute value" column references an operation parameter, in this case an operation name, and the meaning of including that value is listed in the right column.

Table 5 — Required values of OperationsMetadata section attributes

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

Table 6 — Optional values of OperationsMetadata section attributes

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

In addition to the optional values listed in Table 6, there are many optional values of the "name" attributes and "value" elements in the OperationsMetadata section, which may be included when considered useful. Most of these attributes and elements are for recording the domains of various parameters and quantities.

EXAMPLE 1 The domain of the exceptionCode parameter could record all the codes implemented for each operation by that specific server. Similarly, each of the GetCapabilities operation optional request parameters might have its domain recorded.

EXAMPLE 2 The domain of the Sections parameter in the GetCapabilities operation request could record all the sections implemented by that specific server.

12.3.3 Contents section

The Contents section of a service metadata document contains metadata about the data served by this server. For the SPS, this Contents section shall contain information about the sensors that can be tasked and the phenomena that can be measured by these sensors. The following questions shall be answered in the contents section:

- Which sensors can be tasked by the service?
- Which phenomena can be measured with these sensors?
- In which position or region are the sensors operating in or can be tasked to operate?
- Where from can a detailed description of each sensor be received?
- What ID has to be used when invoking operations for a sensor at this service?

A SPS supports the discovery of itself through a registry by two different views. A registry could identify suitable SPSs by either searching the capabilities for a certain type of phenomenon (that can be sensed by at least one sensor managed by the SPS under investigation) in a certain target-area or by searching for sensors with a certain ID and / or certain characteristics which are able to sense a phenomenon in a certain target-area.

The structure of the contents section is given by spsContents.xsd which can be found in annex B and is shown in Figure 17. Following constraints apply for each contents-instance which can be controlled by a combination of unique/key/keyref-elements declared in XMLSchema:

- All SensorOfferings must have different SensorIDs.
- All PhenomenonOfferings must have different Phenomena.
- Each Phenomenon referenced by a SensorOffering must also be declared in a PhenomenonOffering.
- Each SensorID referenced by a PhenomenonOffering must be declared in a SensorOffering.

- There may not be two identical SensorIDs in the same PhenomenonOffering.

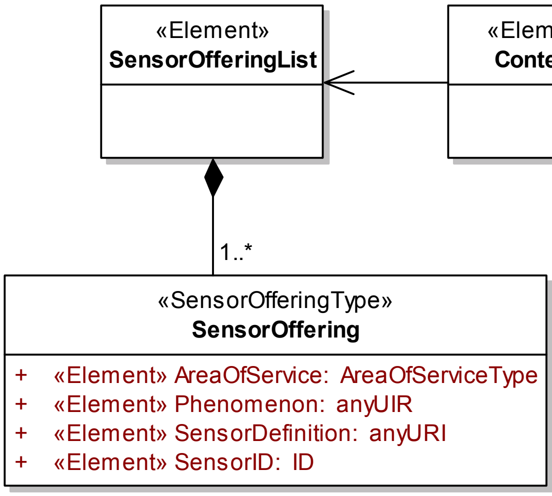


Figure 17: contens section in UML notation (normative), further constraints apply

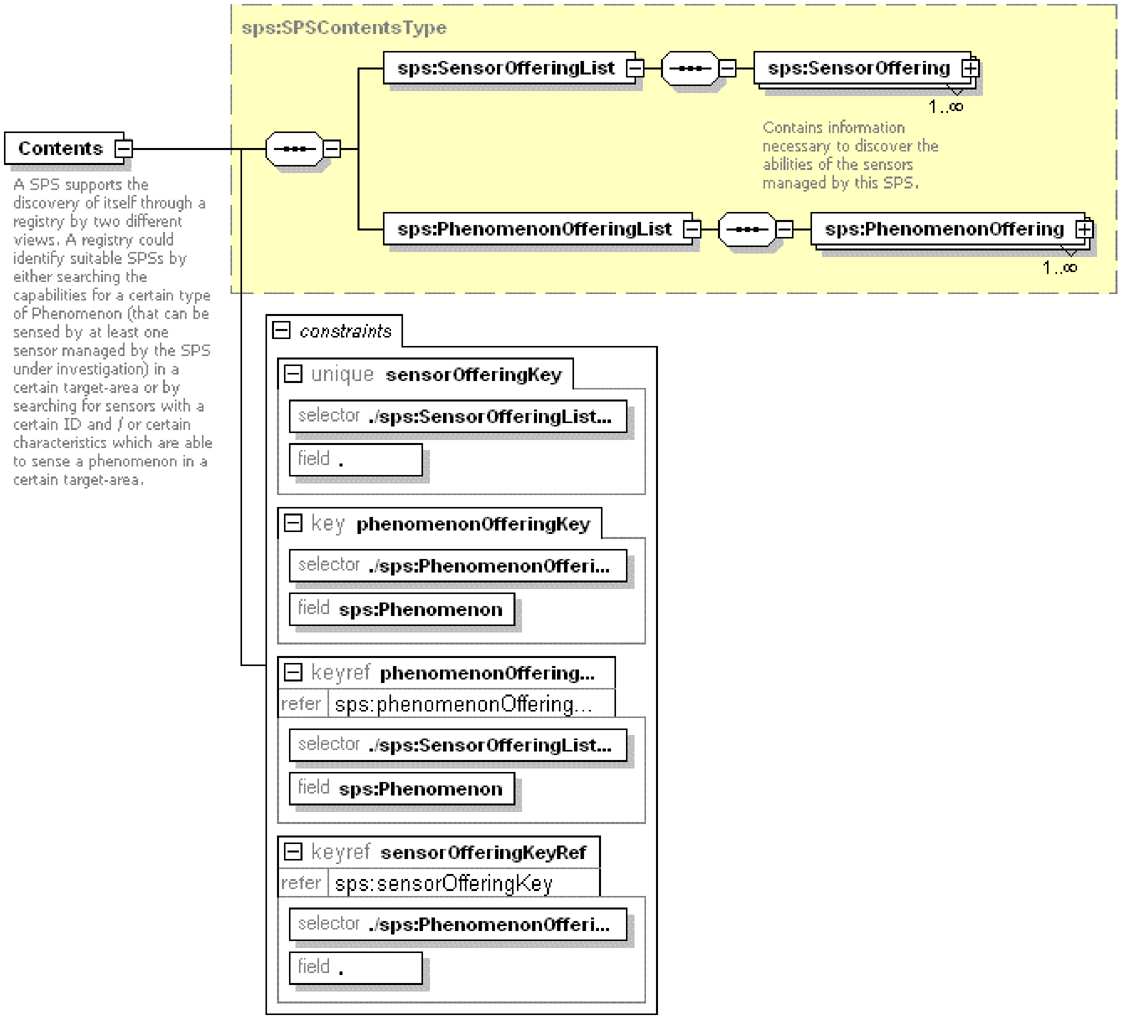


Figure 18: contents section in XMLSpy notation (informative)

12.3.4 Capabilities document XML encoding

A XML schema fragment for a SPS service metadata document extends ows:CapabilitiesBaseType in owsCommon.xsd of [OGC 05-008] and can be found in spsGetCapabilities.xsd in annex B.

This XML Schema Document uses the owsServiceIdentification.xsd, owsServiceProvider.xsd, and owsOperationsMetadata.xsd schemas specified in [OGC 05-008]. It also uses an XML Schema Document for the “Contents” section of the SPS Capabilities XML document, which shall be as shown in Figure 21 and spsContents.xsd file attached in annex B. All these XML Schema Documents contain documentation of the meaning of each element, attribute, and type, and this documentation shall be considered normative as specified in subclause 11.6.3 of [OGC 05-008].

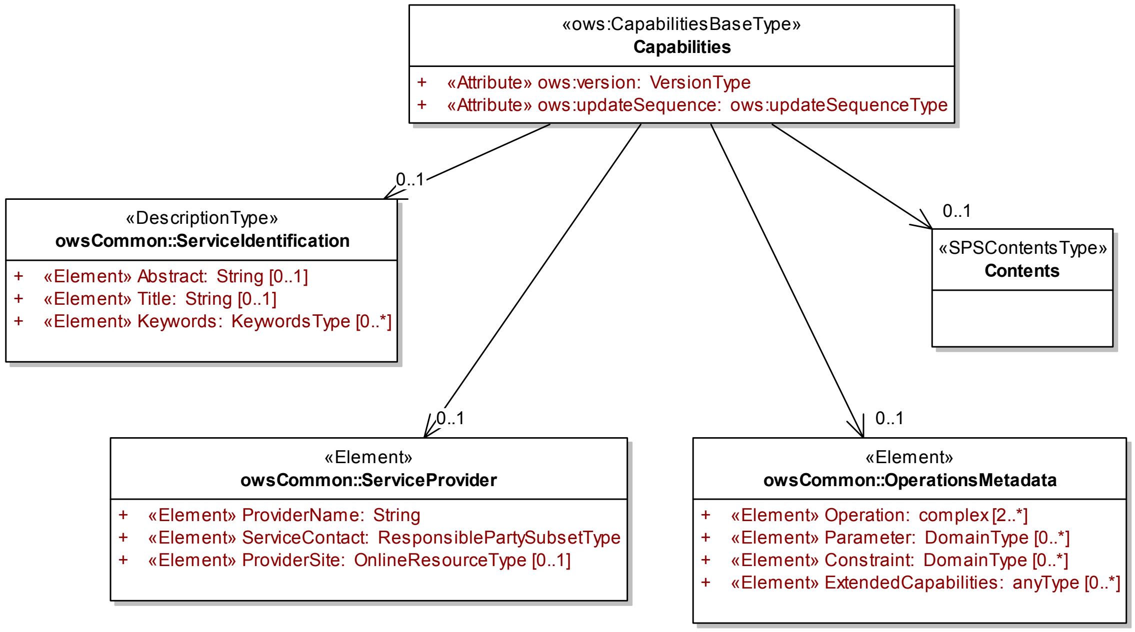


Figure 19: GetCapabilitiesRequestResponse in UML notation (normative)

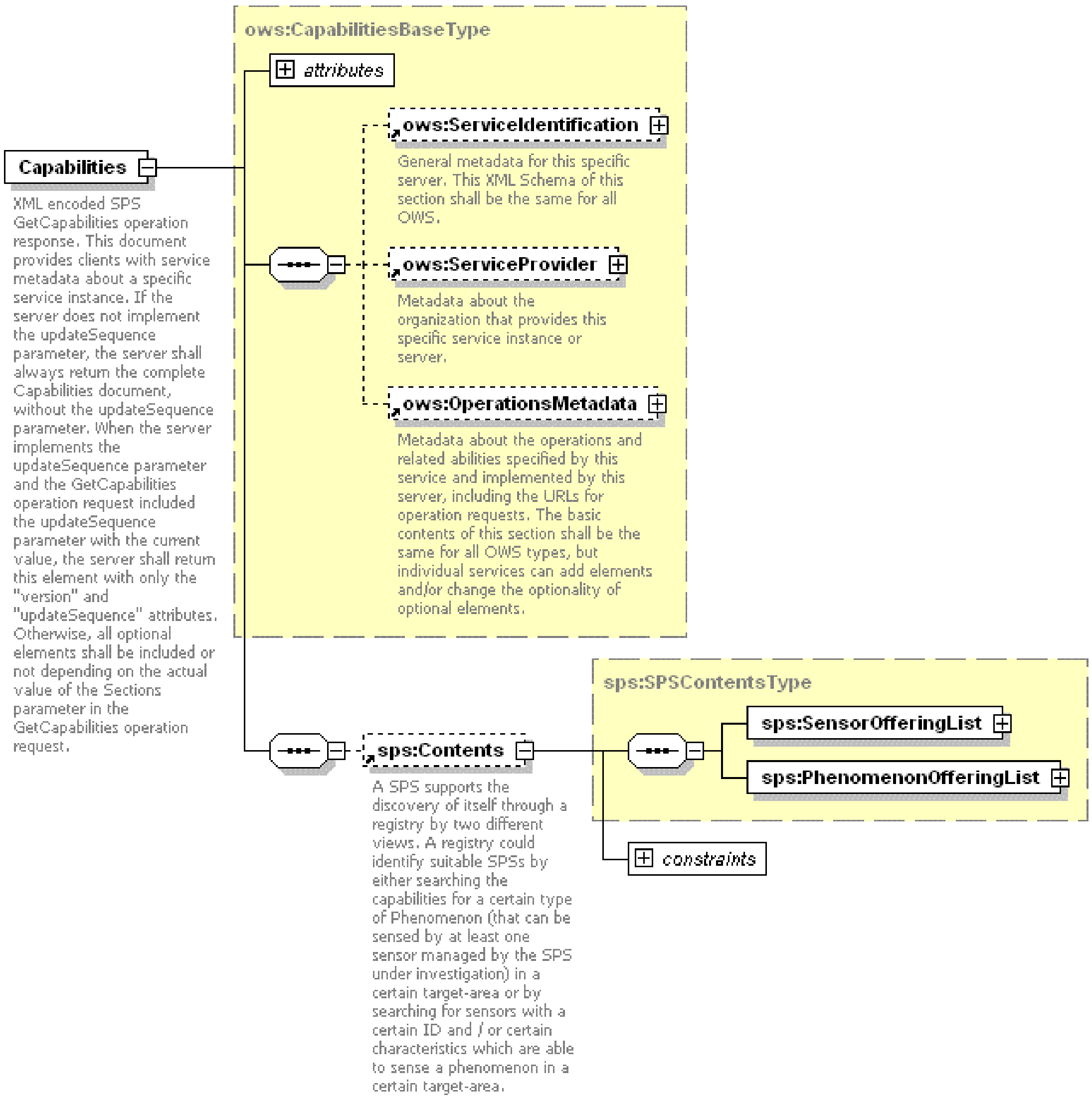


Figure 20: GetCapabilitiesRequestResponse in XMLSpy notation (informative)

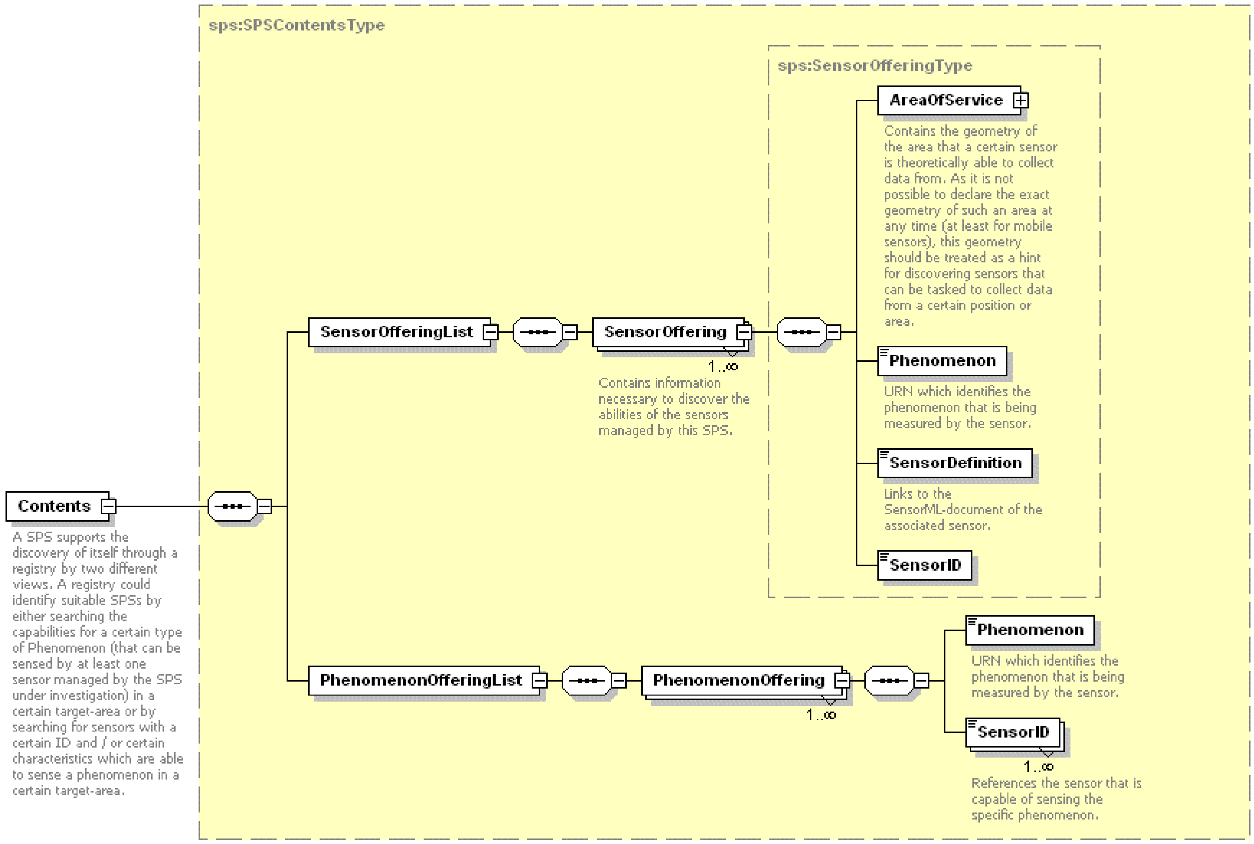


Figure 21: GCRR Contents-section in XMLSpy notation

12.3.5 Capabilities document example

In response to GetCapabilities operation request, a SPS server might generate a document that looks like GCRR.xml in annex D.

12.3.6 Exceptions

When a SPS server encounters an error while performing a GetCapabilities operation, it shall return an exception report message as specified in Clause 8 of [OGC 05-008]. The allowed exception codes shall include those listed in Table 5 of Subclause 7.4.1 of [OGC 05-008], assuming the updateSequence parameter is implemented by the server.

NOTE To reduce the need for readers to refer to other documents, the first six values listed below are copied from Table 20 in Subclause 8.3 of [OGC 05-008].

Table 7 — Exception codes for GetCapabilities operation

exceptionCode value	Meaning of code	“locator” value
OperationNotSupported	Request is for an operation that is not supported by this server	Name of operation not supported

MissingParameterValue	Operation request does not include a parameter value, and this server did not declare a default value for that parameter	Name of missing parameter
InvalidParameterValue	Operation request contains an invalid parameter value	Name of parameter with invalid value
VersionNegotiationFailed	List of versions in 'AcceptVersions' parameter value in GetCapabilities operation request did not include any version supported by this server	None, omit 'locator' parameter
InvalidUpdateSequence	Value of (optional) updateSequence parameter in GetCapabilities operation request is greater than current value of service metadata updateSequence number	None, omit 'locator' parameter
NoApplicableCode	No other exceptionCode specified by this service and server applies to this exception	None, omit “locator” parameter
InvalidRequest	The request a client sent to initiate an operation does not conform to the schema for this operation	Error message from the schema-validator

13 DescribeTasking operation (mandatory)

13.1 Introduction

The DescribeTasking operation request allows SPS clients to request the information that is needed in order to prepare a tasking request targeted at the assets that are supported by the SPS and that are selected by the client. The server will return information about all parameters that have to be set by the client in order to perform a Submit operation. The only additional parameter “SensorID” defines the specific sensor(s) that shall be described by the server. This allows servers to façade multiple assets that require parameterization and return all information to the client using one call only.

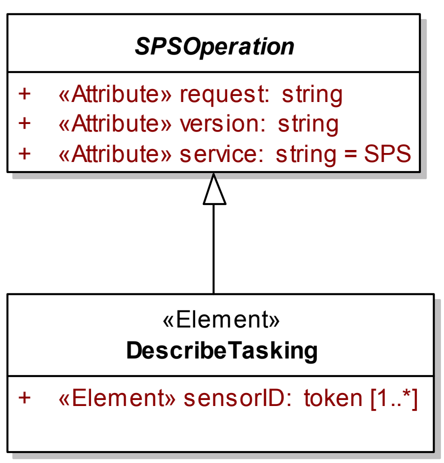


Figure 22: DescribeTasking in UML notation

13.2 DescribeTasking operation request

13.2.1 DescribeTasking request parameters

This table also specifies the UML model data type, source of values, and multiplicity of each listed parameter, plus the meaning to servers when each optional parameter is not included in the operation request.

Table 8: Parameters in DescribeTasking operation request

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

NOTE 2 The data type of many parameters is specified as “Character String type, not empty”. In the XML Schema Documents specified herein, these parameters are encoded with the xsd:string type, which does NOT require that these strings not be empty.

The “Multiplicity and use” columns in Table 7 through all tables specify the optionality of each listed parameter and data structure in the DescribeTasking operation request. Since all parameters and data structures are mandatory in the operation request, all parameters and data structures shall be implemented by all SPS clients, using a specified value(s). Similarly, all parameters and data structures shall be implemented by all SPS servers, checking that each request parameter is received with any specified value(s).

13.2.2 DescribeTasking request KVP encoding

```
KVP encoding not supported
```

13.2.3 DescribeTasking request XML encoding

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

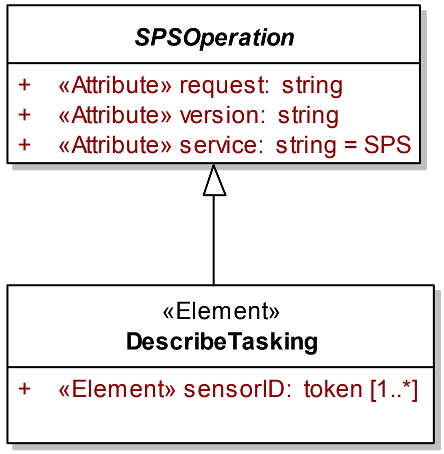


Figure 23: DescribeTasking in UML notation

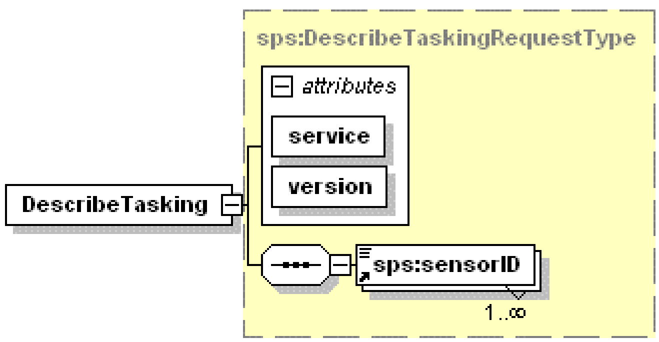


Figure 24: DescribeTakingRequest in XMLSpy notation

EXAMPLE An example DescribeTasking operation request XML encoded for HTTP POST is:

```
<?xml version="1.0" encoding="UTF-8"?>
<DescribeTasking xmlns="http://www.opengis.net/sps" service="SPS"
version="1.0.0">
  <sensorID>urn:x-ogc:object:sensor:IFGI:AXISPTZ_C:1:ifgicam01</sensorI
D>
</DescribeTasking>
```

13.3 DescribeTasking operation response

If the client provides one to many valid sensorIDs within the DescribeTasking, the SPS server will respond with a DescribeTaskingRequestResponse. The necessary parameters for every sensor will be described in a “taskingDescriptor”. This element contains a “sensorID” element that identifies the sensor for which the descriptor(s) applie(s). The descriptor itself is described in subclause 11.2.1. The following figures illustrate the response.

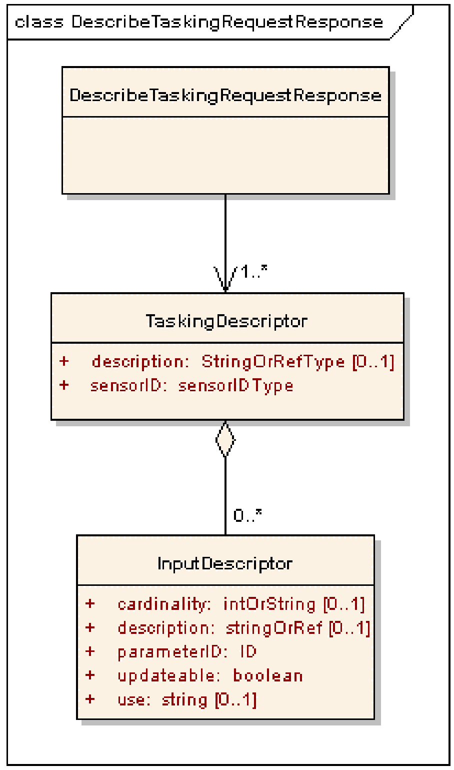


Figure 25: DescribeTaskingRequestResponse in UML notation

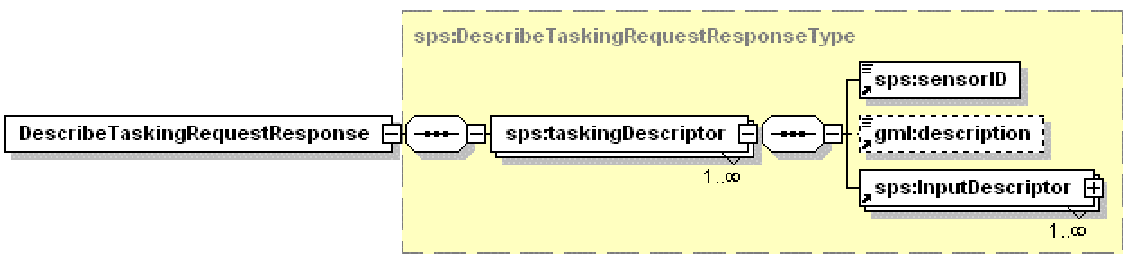


Figure 26: DescribeTaskingRequestResponse in XMLSpy notation

13.3.1 Normal response parameters

The normal response to a valid DescribeTasking operation request shall be a DescribeTaskingRequestResponse. More precisely, a response from the DescribeTasking operation shall include the parts listed in Table 9. This table also specifies the UML model data type plus the multiplicity and use of each listed part.

Table 9 — Parts of DescribeTasking operation response

Name	Definition	Data type and values	Multiplicity and use
taskingDescriptor	Provides all information necessary to task an asset	complex type	one to many, mandatory

13.3.2 Normal response XML encoding

The following schema fragment specifies the contents and structure of a DescribeTasking operation response, always encoded in XML:

See annex B.

13.3.3 DescribeTasking response example

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

See Running Example, chapter 20.

13.3.4 DescribeTasking exceptions

When a SPS server encounters an error while performing a DescribeTasking operation, it shall return an exception report message as specified in Subclause 7.4 of [OGC 05-008]. The allowed standard exception codes shall include those listed in Table 10. For each listed exceptionCode, the contents of the "locator" parameter value shall be as specified in the right column.

Table 10: Exception codes for DescribeTasking operation

exceptionCode value	Meaning of code	"locator" value
OperationNotSupported	Request is for an operation that is not supported by this server	Name of operation not supported
MissingParameterValue	Operation request does not include a parameter value, and this server did not declare a default value for that parameter	Name of missing parameter
InvalidParameterValue	Operation request contains an invalid parameter value	Name of parameter with invalid value
UnknownSensorID	SensorID that has been issued by the client is not known to the SPS	(comma-separated list of) the unknown sensorID(s)
NoApplicableCode	No other exceptionCode specified by this service and server applies to this exception	None, omit "locator" parameter
InvalidRequest	Request does not conform to the schema for this operation	Exception message generated by validator

14 GetFeasibility operation (optional)

14.1 Introduction

The GetFeasibility operation allows SPS clients to obtain information about the feasibility of a tasking request. Dependent on the types of assets façaded by the SPS, the SPS server action may be as simple as checking that the request parameters are valid, and are consistent with certain business rules, or it may be a complex operation that calculates the utilizability of the asset to perform a specific task at the defined location, time, orientation, calibration etc.

14.2 GetFeasibility operation

The following figures illustrate the GetFeasibility operation parameters in UML and XMLSpy notation.

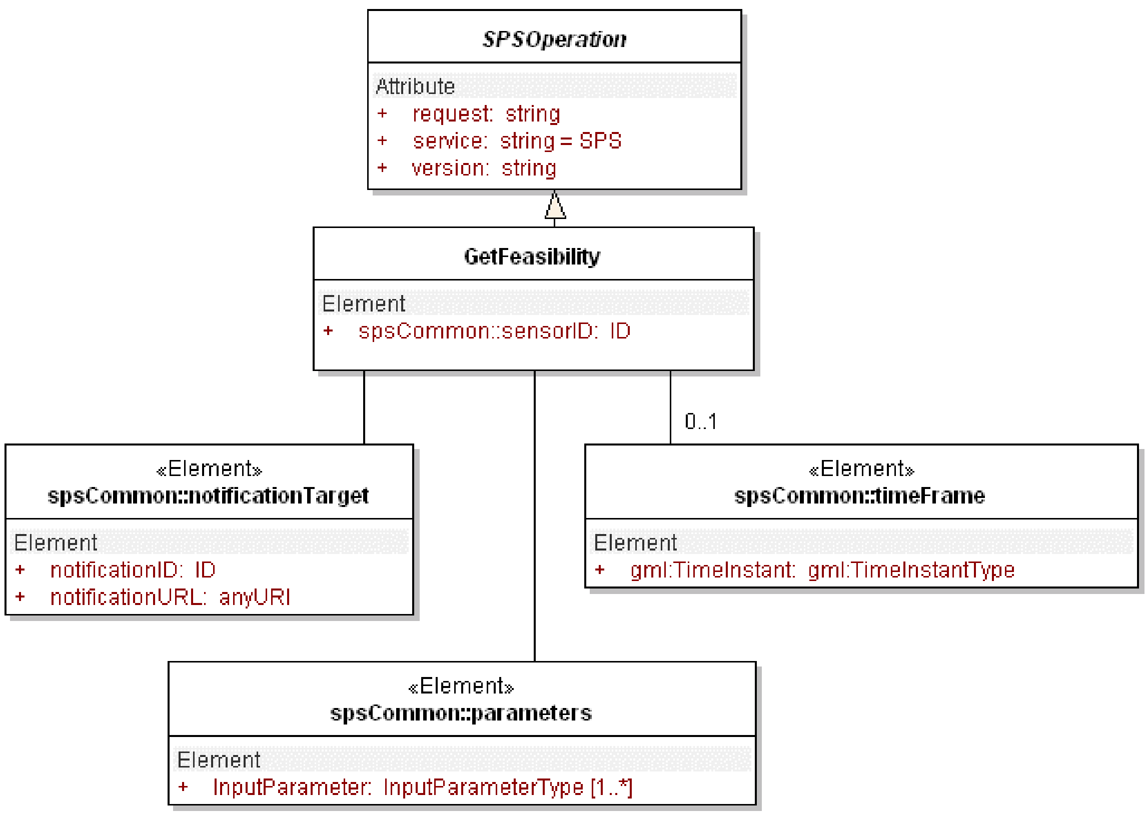


Figure 27: GetFeasibilityRequest in UML notation

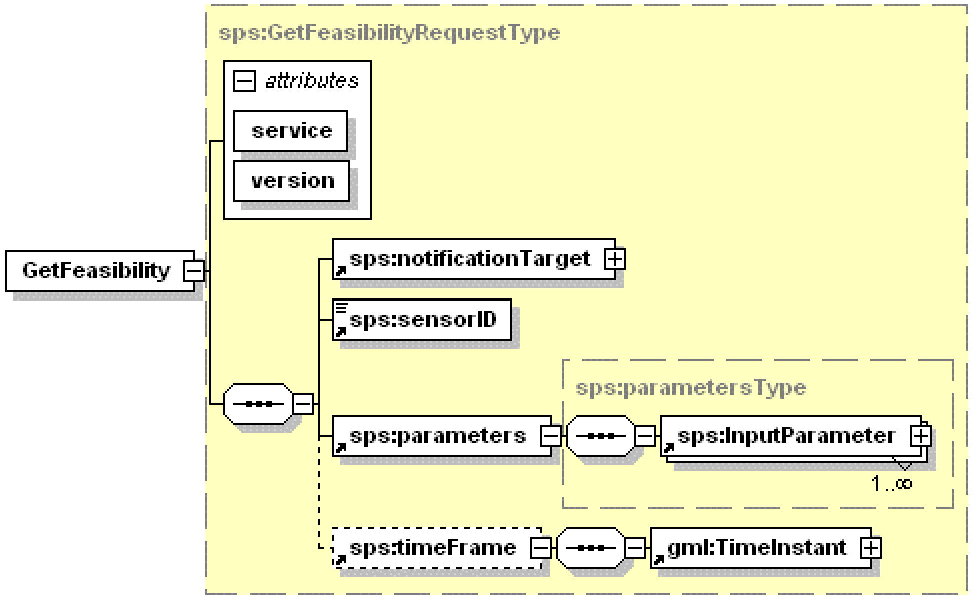


Figure 28: GFR in XMLSpy notation

14.2.1 GetFeasibility operation parameters

This table also specifies the UML model data type, source of values, and multiplicity of each listed parameter, plus the meaning to servers when each optional parameter is not included in the operation request. Although some values listed in the "Name" column appear to contain spaces, they shall not contain spaces.

Table 11 — Parameters in GetFeasibility operation request

Name [a]	Definition	Data type and values	Multiplicity and use
service	Service type identifier	Character String type, not empty Value is “SPS”	One (mandatory)
request	Operation name	Character String type, not empty Value is “GetFeasibility”	One (mandatory)
version	Specification version for operation	Character String type, not empty Value is same as version of this document	One (mandatory)
notification Target	Defines the WNS that has to be used to notify the client about the request results	complex type	one (mandatory)
sensorID	Identifies the sensor that shall be tasked	ID	one (mandatory)
parameters	Input parameter values. Encoding should match description in DescribeTaskingRequest Response	complex	one (mandatory)
timeFrame	Maximum point in time the request keeps valid.	gml:TimeInstant	one (optional)
a The name capitalization rules being used here are specified in Subclause 11.6.2 of [OGC 05-008].			

NOTE 2 The data type of many parameters is specified as “Character String type, not empty”. In the XML Schema Documents specified herein, these parameters are encoded with the xsd:string type, which does NOT require that these strings not be empty.

The “Multiplicity and use” columns in Table 7 through all tables specify the optionality of each listed parameter and data structure in the GetFeasibilityRequest operation. Since all parameters and data structures are mandatory in the operation request, all parameters and data structures shall be implemented by all SPS clients, using a specified value(s). Similarly, all parameters and data structures shall be implemented by all SPS servers, checking that each request parameter is received with any specified value(s).

14.2.2 GetFeasibility request KVP encoding

No HTTP GET supported.

14.2.3 GetFeasibility request XML encoding

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

See annex B.

EXAMPLE An example GetFeasibility operation request XML encoded for HTTP POST is:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<GetFeasibility xmlns="http://www.opengis.net/sps"
xmlns:gml="http://www.opengis.net/gml"
xmlns:swe="http://www.opengis.net/swe/0" service="SPS" version="1.0.0">
   <notificationTarget>
      <notificationID>1234</notificationID>
      <notificationURL>http://mars.uni-
muenster.de:8080/WNS/wns</notificationURL>
   </notificationTarget>
   <sensorID>urn:x-ogc:object:sensor:IFGI:AXISPTZ_C:1:ifgicam01</sensorI
D>
   <parameters>
      <InputParameter parameterID="task-start-time">
         <value>
            <swe:Time>2005-10-05T16:25:00+02:00</swe:Time>
         </value>
      </InputParameter>
      <InputParameter parameterID="task-end-time">
         <value>
            <swe:Time>2005-10-05T16:55:00+02:00</swe:Time>
         </value>
      </InputParameter>
   </parameters>
   <timeFrame>
      <gml:TimeInstant>
         <gml:timePosition>2005-10-05T16:20:00</gml:timePosition>
      </gml:TimeInstant>
   </timeFrame>
</GetFeasibility>
```

14.3 GetFeasibility operation response

14.3.1 Normal response parameters

The normal response to a valid GetFeasibility operation shall be GetFeasibilityRequestResponse. More precisely, a response from the GetFeasibility operation shall include the parts listed in Table 12 and illustrated in the following figures. This table also specifies the UML model data type plus the multiplicity and use of each listed part.

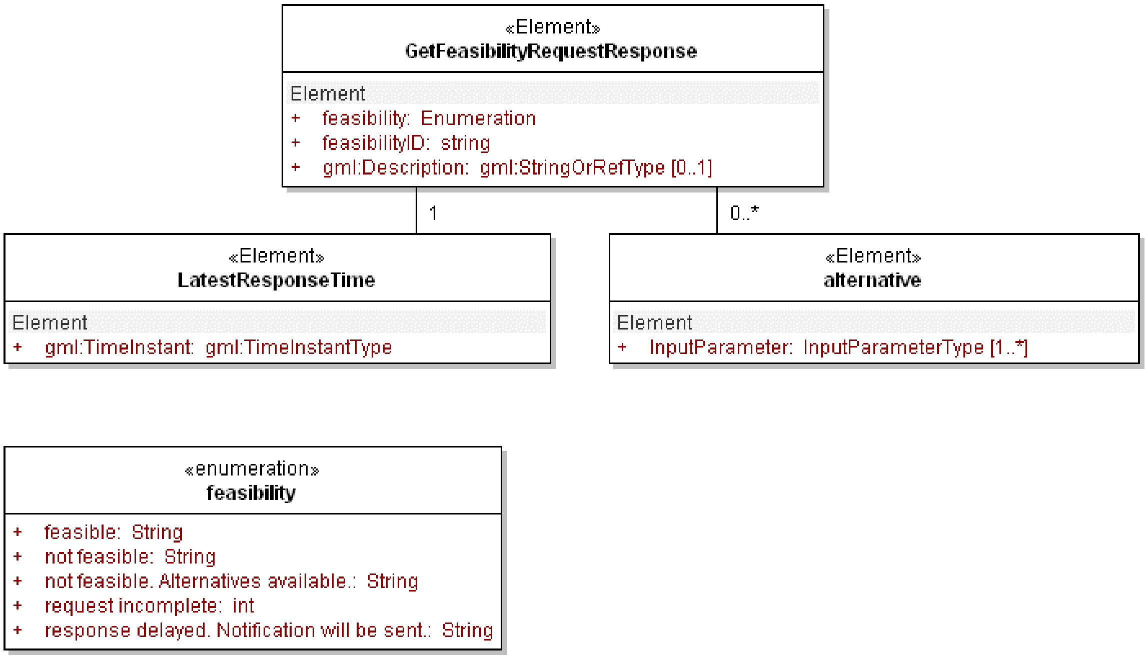


Figure 29: GetFeasibilityRequestResponse in UML notation

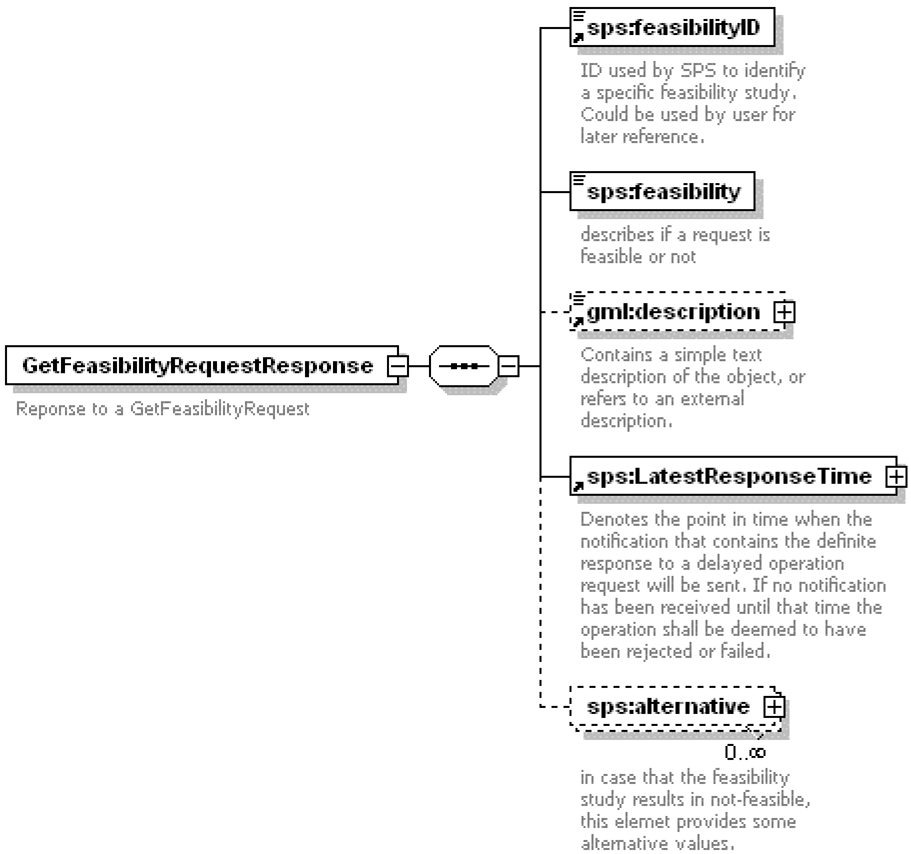


Figure 30: GFRR in XMLSpy notation

Table 12: Parts of GetFeasibility operation response

Name	Definition	Data type and values	Multiplicity and use
feasibilityID	Identifier for the feasibility study	ID	one (mandatory)
feasibility	Identifier for the feasibility status	String, enumerates: "feasible" "not feasible" "response delayed, notification will be sent" "request incomplete" "not feasible, alternatives available"	one (mandatory)
description	Text description of the reponse	StringOrRefType	one (optional)
LatestResponse Time	GetFeasibility response will be sent until LatestResponseTime at latest. In case that no response is received, the operation shall be evaluated as non-feasible.	gml:TimeInstantType	one (mandatory)
alternative	Possible alternative to a given set of parameters will lead to status "not feasible"	InputParameterType	one to many (optional)

14.3.2 Normal response XML encoding

See annex B.

14.3.3 GetFeasibility response example

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

```xml
<?xml version="1.0" encoding="UTF-8"?>
<GetFeasibilityRequestResponse xmlns="http://www.opengis.net/sps"
xmlns:gml="http://www.opengis.net/gml">
   <feasibilityID>17</feasibilityID>
   <feasibility>feasible</feasibility>
   <LatestResponseTime>
      <gml:TimeInstant>
         <gml:timePosition>2005-10-05T16:10:00</gml:timePosition>
      </gml:TimeInstant>
   </LatestResponseTime>
</GetFeasibilityRequestResponse>
```

14.3.4 GetFeasibility exceptions

When a SPS server encounters an error while performing a GetFeasibility operation, it shall return an exception report message as specified in Subclause 7.4 of [OGC 05-008]. The allowed standard exception codes shall include those listed in Table 13. For each listed exceptionCode, the contents of the “locator” parameter value shall be as specified in the right column.

NOTE To reduce the need for readers to refer to other documents, the first four values listed below are copied from Table 20 in Subclause 8.3 of [OGC 05-008].

Table 13 — Exception codes for GetFeasibility operation

exceptionCode value	Meaning of code	“locator” value
OperationNotSupported	Request is for an operation that is not supported by this server	Name of operation not supported
MissingParameterValue	Operation request does not include a parameter value, and this server did not declare a default value for that parameter	Name of missing parameter
InvalidParameterValue	Operation request contains an invalid parameter value	Name of parameter with invalid value
NoApplicableCode	No other exceptionCode specified by this service and server applies to this exception	None, omit “locator” parameter
InvalidRequest	Request is not conform to the schema for this operation	Exception message generated by validator
UnknownSensorID	SensorID that has been issued by the client is not known to the SPS	(comma-separated list of) the unknown sensorID(s)

15 Submit operation (mandatory)

15.1 Introduction

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

15.2 Submit operation request

The following figures illustrate the Submit operation parameters in UML and XMLSpy notation.

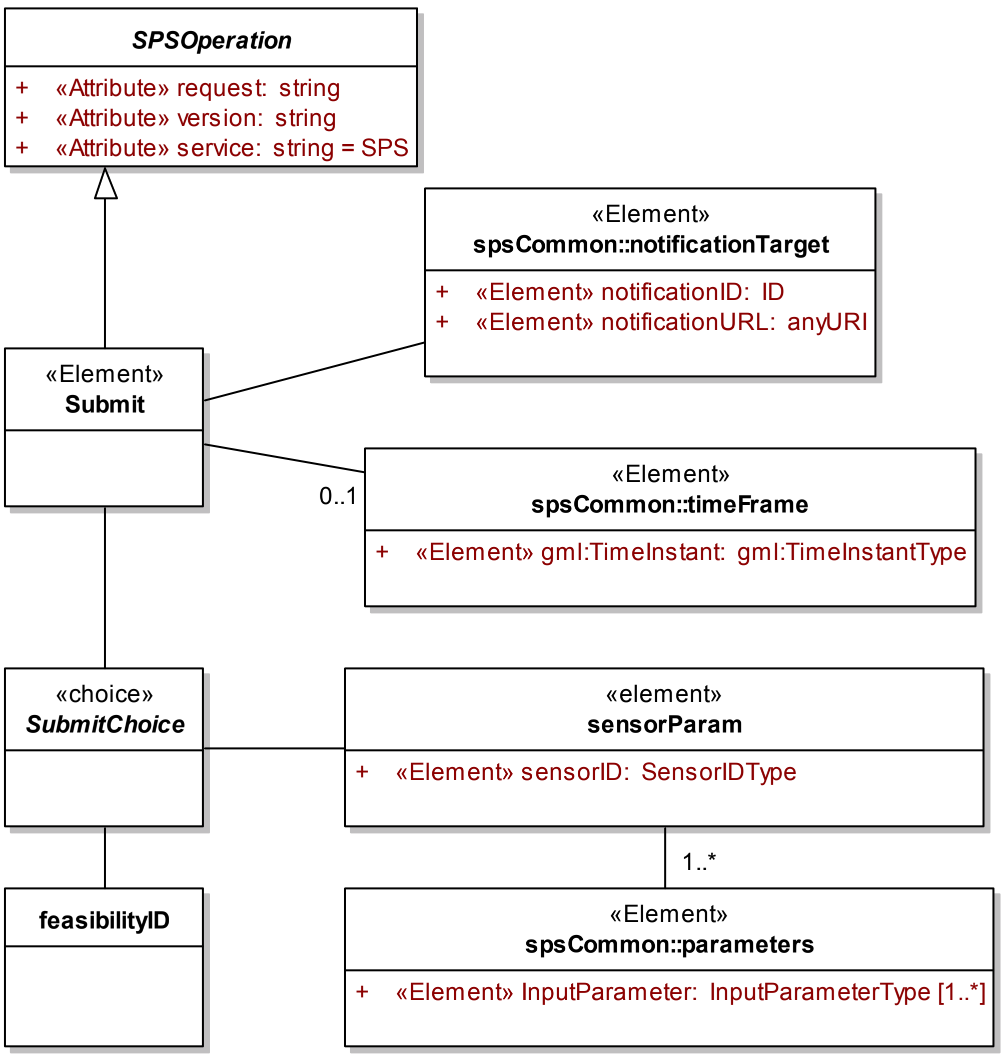


Figure 31: Submit in UML notation

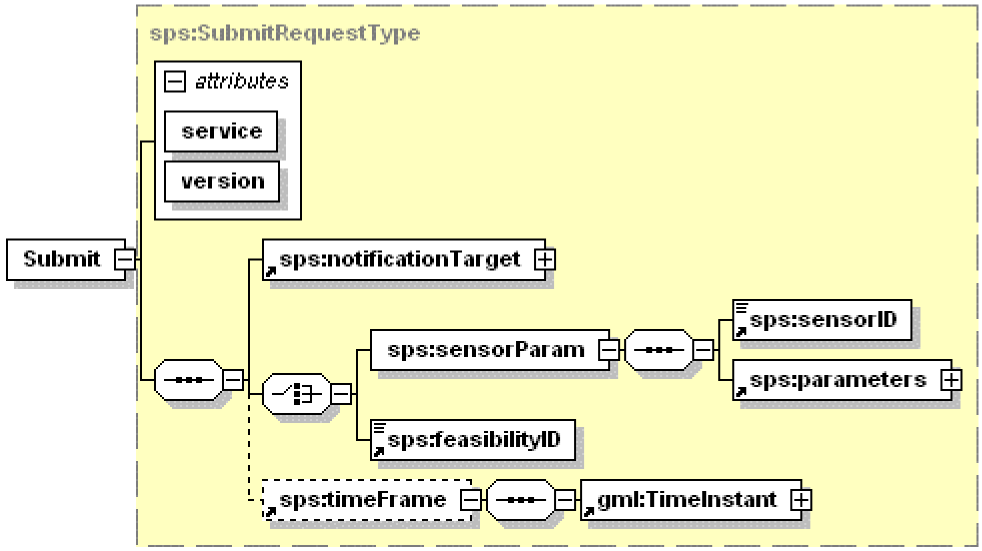


Figure 32: Submit in XMLSpy notation

15.2.1 Submit request parameters

This table also specifies the UML model data type, source of values, and multiplicity of each listed parameter, plus the meaning to servers when each optional parameter is not included in the operation request. Although some values listed in the "Name" column appear to contain spaces, they shall not contain spaces.

Table 14 — Parameters in Submit operation

Name [a]	Definition	Data type and values	Multiplicity and use
service	Service type identifier	Character String type, not empty Value is "SPS"	One (mandatory)
request	Operation name	Character String type, not empty Value is operation name "Submit"	One (mandatory)
version	Specification version for operation	Character String type, not empty Value is equal to the number of this document.	One (mandatory)
notification Target	see subclause 11.2.3	see subclause 11.2.3	one (mandatory)
sensorParam	container element contains sensorID and parameters elements	complex	one (mandatory) only instead of feasibilityID
feasibilityID	Identifier of the feasibility study ID that was provided by SPS on behalf of a GetFeasibility request	String	one (mandatory) only instead of sensorParam element
timeFrame	defines the maximum point in time the request keeps valid	gml:TimeInstant	one (optional)
a The name capitalization rules being used here are specified in Subclause 11.6.2 of [OGC 05-008].			

NOTE 2 The data type of many parameters is specified as "Character String type, not empty". In the XML Schema Documents specified herein, these parameters are encoded with the xsd:string type, which does NOT require that these strings not be empty.

The "Multiplicity and use" columns in all tables specify the optionality of each listed parameter and data structure in the Submit operation. All the "mandatory" parameters and data structures shall be implemented by all SPS clients, using a specified value(s). Similarly, all the "mandatory" parameters and data structures shall be implemented by all SPS servers, checking that each request parameter or data structure is received with any specified value(s).

All the "optional" parameters and data structures, in the Submit operation request, should be implemented by all SPS clients using specified values, for each implemented process to which that parameter or data structure applies. Similarly, all the "optional" parameters and data structures shall be implemented by all SPS servers, for each implemented process to which that parameter or data structure applies.

15.2.2 Submit request KVP encoding

KVP encoding not supported.

15.2.3 Submit request XML encoding

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

See annex B.

EXAMPLE An example Submit operation request XML encoded for HTTP POST is:

```
<?xml version="1.0" encoding="UTF-8"?>
<Submit xmlns="http://www.opengis.net/sps"
xmlns:gml="http://www.opengis.net/gml"
xmlns:swe="http://www.opengis.net/swe/0" service="SPS" version="1.0.0">
   <notificationTarget>
      <notificationID>1234</notificationID>
      <notificationURL>http://mars.uni-
muenster.de:8080/WNS/wns</notificationURL>
   </notificationTarget>
   <sensorParam>

   <sensorID>urn:x-ogc:object:sensor:IFGI:AXISPTZ_C:1:ifgicam01</sensorI
D>
      <parameters>
         <InputParameter parameterID="task-start-time">
            <value>
               <swe:Time>2005-10-05T16:25:00+02:00</swe:Time>
            </value>
         </InputParameter>
         <InputParameter parameterID="task-end-time">
            <value>
               <swe:Time>2005-10-05T16:30:00+02:00</swe:Time>
            </value>
         </InputParameter>
      </parameters>
   </sensorParam>
   <timeFrame>
      <gml:TimeInstant>
         <gml:timePosition>2005-10-05T12:00:00</gml:timePosition>
      </gml:TimeInstant>
   </timeFrame>
</Submit>
```

15.3 Submit operation response

15.3.1 Normal response parameters

The normal response to a valid Submit operation request shall be SubmitRequestResponse. More precisely, a response from the Submit operation shall include the parts listed in Table 15 and illustrated in the following figures. This table also specifies the UML model data type plus the multiplicity and use of each listed part.

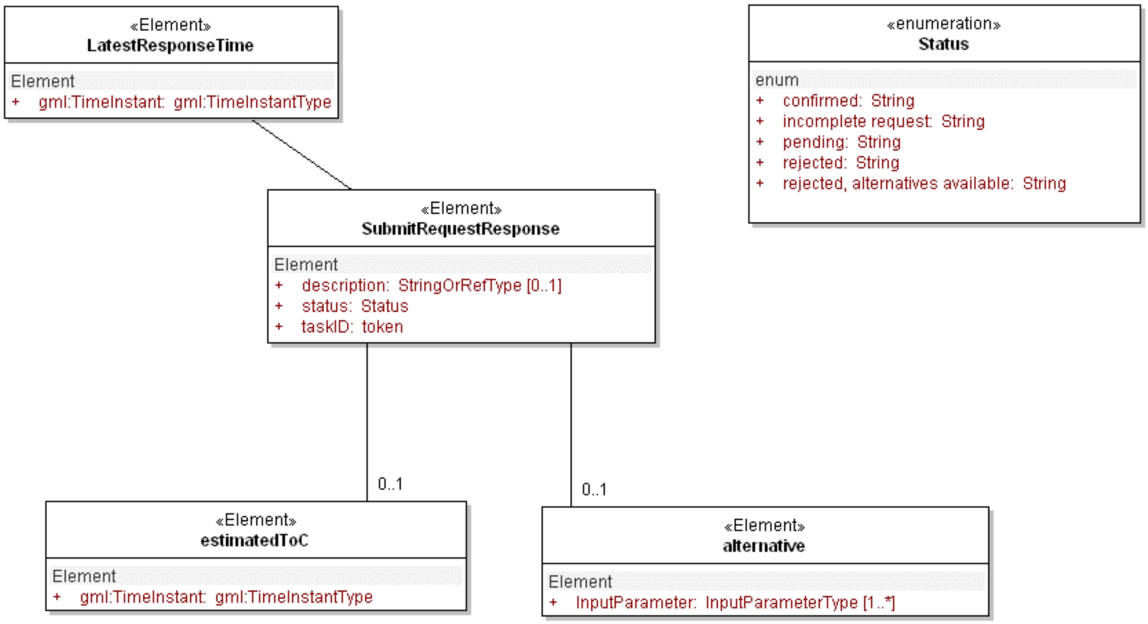


Figure 33: SubmitRequestResponse parameter in UML notation

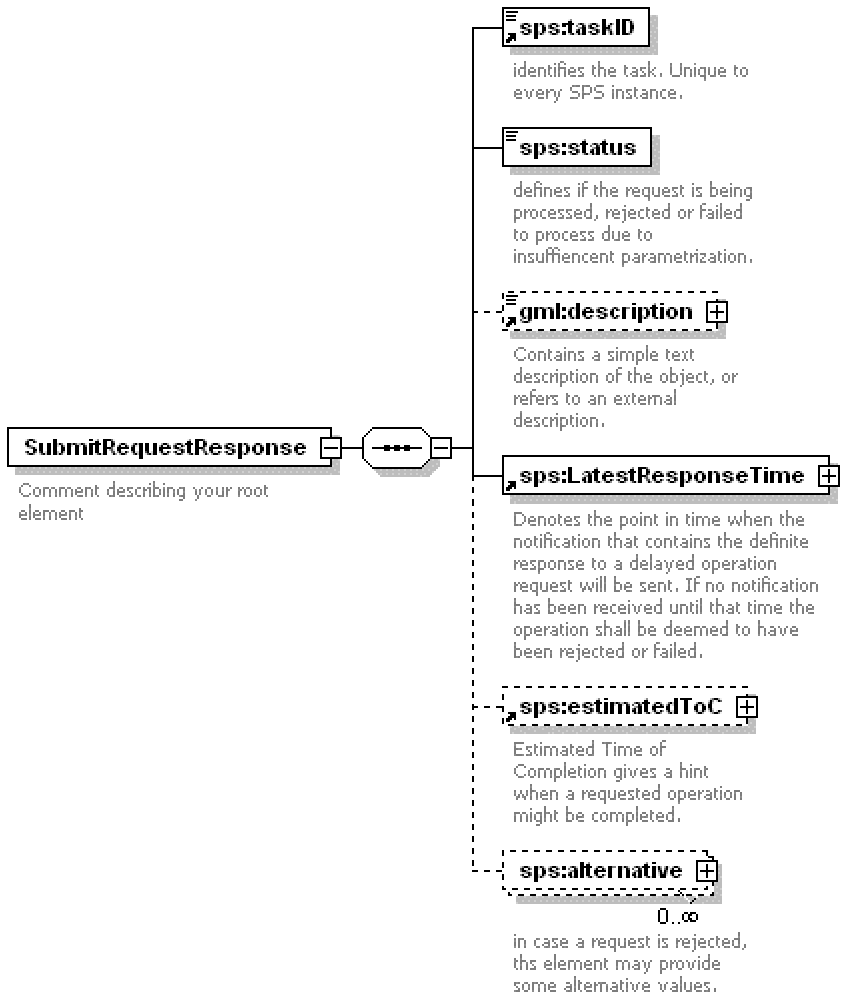


Figure 34: SubmitRequestResponse parameter in XMLSpy notation

Table 15 — Parts of Submit operation response

Name	Definition	Data type and values	Multiplicity and use
taskID	Identifier for this task, needed for subsequent update requests	token	one (mandatory)
status	Identifier of the status of the request	String enumerates: “confirmed” “rejected” “incomplete request” “pending” “rejected, alternatives available”	one (mandatory)
description	Additional metadata	String or reference to external source	one (optional)
estimatedToC	Defines estimated time of completion	gml:TimeInstant	one (optional)
alternative	Provides alternatives if the sensors are not taskable as requested	complex: InputParameter	one to many (optional)
LatestResponseTime	Submit response will be sent until LatestResponseTime at latest. In case that no response is received, the operation shall be evaluated as rejected.	gml:TimeInstantType	one (mandatory)
a			

15.3.2 Normal response XML encoding

The following schema fragment specifies the contents and structure of a Submit operation response, always encoded in XML.

See annex B.

15.3.3 Submit response example

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

```
<?xml version="1.0" encoding="UTF-8"?>
<SubmitRequestResponse xmlns="http://www.opengis.net/sps"
xmlns:gml="http://www.opengis.net/gml">
  <taskID>433</taskID>
  <status>confirmed</status>
  <LatestResponseTime>
    <gml:TimeInstant>
      <gml:timePosition>2005-10-05T16:15:00</gml:timePosition>
    </gml:TimeInstant>
  </LatestResponseTime>
```



```
   <estimatedToC>
      <gml:TimeInstant>
         <gml:timePosition>2005-10-05T16:55:00+02:00</gml:timePosition>
      </gml:TimeInstant>
   </estimatedToC>
</SubmitRequestResponse>
```

15.3.4 SubmitRequestResponse exceptions

When a SPS server encounters an error while performing a Submit operation, it shall return an exception report message as specified in Subclause 7.4 of [OGC 05-008]. The allowed standard exception codes shall include those listed in Table 16. For each listed exceptionCode, the contents of the "locator" parameter value shall be as specified in the right column of Table 16.

Table 16 — Exception codes for Submit operation

exceptionCode value	Meaning of code	"locator" value
OperationNotSupported	Request is for an operation that is not supported by this server	Name of operation not supported
MissingParameterValue	Operation request does not include a parameter value, and this server did not declare a default value for that parameter	Name of missing parameter
InvalidParameterValue	Operation request contains an invalid parameter value	Name of parameter with invalid value
UnknownSensorID	SensorID that has been issued by the client is not known to the SPS	(comma-separated list of) the unknown sensorID(s)
NoApplicableCode	No other exceptionCode specified by this service and server applies to this exception	None, omit "locator" parameter
FeasibilityIDExpired	feasibilityID that has been issued by the client is no longer supported by the service	None, omit 'locator' parameter
InvalidRequest	Request not conform to the schema for this operation	Exception message generated by validator

16 GetStatus operation (optional)

16.1 Introduction

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

16.2 GetStatus operation request

The following figures illustrate the GetStatus operation parameters in UML and XMLSpy notation.

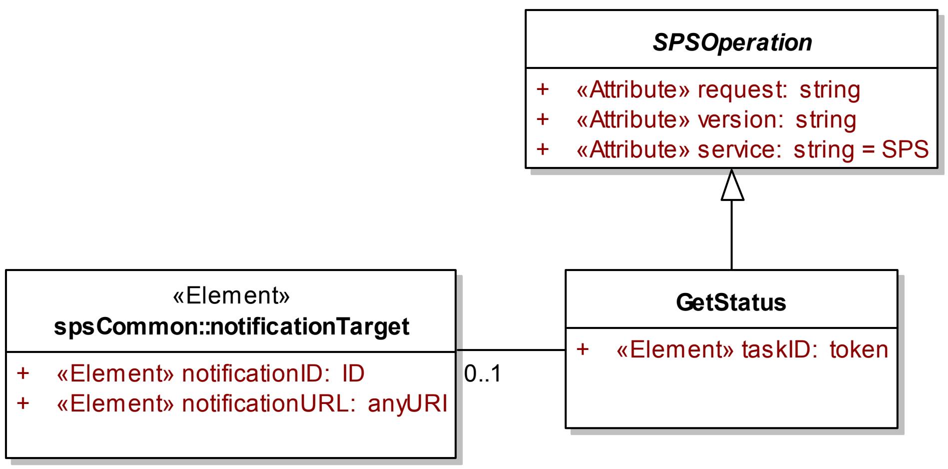


Figure 35: GetStatus operation parameters in UML notation

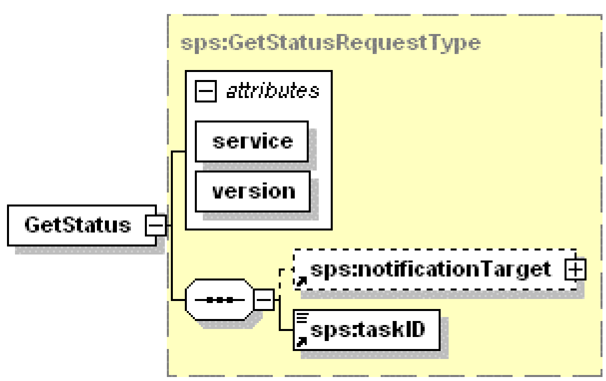


Figure 36: GetStatus operation parameters in XMLSpy notation

16.2.1 GetStatus request parameters

This table also specifies the UML model data type, source of values, and multiplicity of each listed parameter, plus the meaning to servers when each optional parameter is not included in the operation request. Although some values listed in the "Name" column appear to contain spaces, they shall not contain spaces.

Table 17 — Parameters in GetStatus operation

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

NOTE 2 The data type of many parameters is specified as "Character String type, not empty". In the XML Schema Documents specified herein, these parameters are encoded with the xsd:string type, which does NOT require that these strings not be empty.

The "Multiplicity and use" columns in all tables specify the optionality of each listed parameter and data structure in the GetStatus operation. All the "mandatory" parameters and data structures shall be implemented by all SPS clients, using a specified value(s). Similarly, all the "mandatory" parameters and data structures shall be implemented by all SPS servers, checking that each request parameter or data structure is received with any specified value(s).

All the "optional" parameters and data structures, in the GetStatus operation, should be implemented by all SPS clients using specified values, for each implemented process to which that parameter or data structure applies. Similarly, all the "optional" parameters and data structures shall be implemented by all SPS servers, for each implemented process to which that parameter or data structure applies.

16.2.2 GetStatus request KVP encoding

KVP encoding not supported.

16.2.3 GetStatus request XML encoding

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

See annex B.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<GetStatus xmlns="http://www.opengis.net/sps" service="SPS"
version="1.0.0">
   <taskID>433</taskID>
</GetStatus>
```

16.3 GetStatus operation response

16.3.1 Normal response parameters

The normal response to a valid GetStatus operation shall be a GetStatusRequestResponse. More precisely, a response from the GetStatus operation shall include the parts listed in Table 18 and shown in the following figures. This table also specifies the UML model data type plus the multiplicity and use of each listed part.

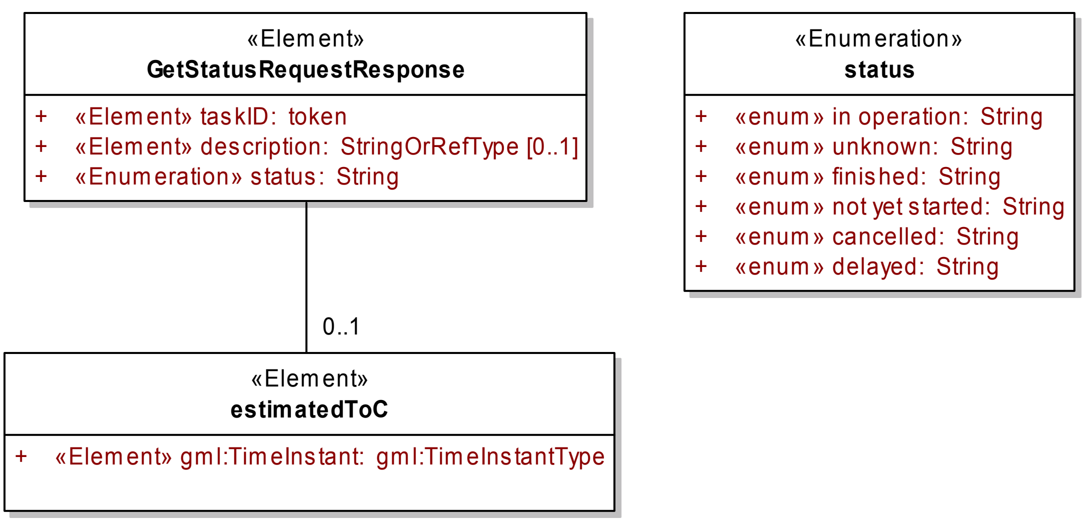


Figure 37: GetStatusRequestResponse parameters in UML notation

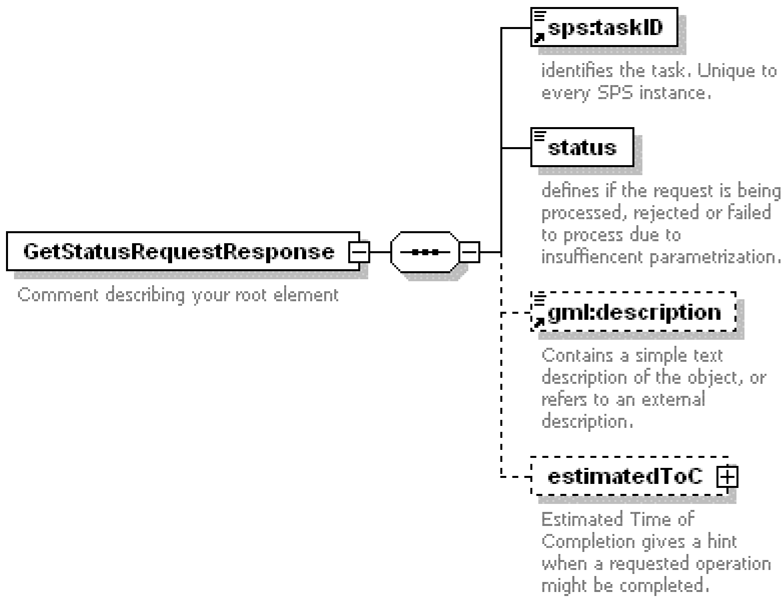


Figure 38: GSRR parameters in XMLSpy notation

Table 18 — Parts of GetStatus operation response

Name	Definition	Data type and values	Multiplicity and use
taskID	Identifier for the requested task	Token	one (mandatory)
status	Defines the current status of this task	String, enumerates: “unknown”, “in operation”, “finished”, “not yet started”, “cancelled”, “delayed”	one (mandatory)
description	Optional further information about the task	String	one (optional) Note: jf the SPS operator cancelled or delayed the task, then the gml:description should be considered as a mandatory element explaining why the task was cancelled or delayed. If the task was delayd by the operator, then the sps:estimatedToC element should, if possible, indicate when the task is supposed to be finished despite the delay.
estimatedToC	Defines the estimated Time of Completion of this task	gml:TimeInstant	one (optional)
a			

16.3.2 Normal response XML encoding

The following schema fragment specifies the contents and structure of a GetStatus operation response, always encoded in XML.

See annex B.

16.3.3 GetStatus response example

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

```
<?xml version="1.0" encoding="UTF-8"?>
<GetStatusRequestResponse xmlns="http://www.opengis.net/sps">
  <taskID>433</taskID>
```



```
   <status>in operation</status>
</GetStatusRequestResponse>
```

16.3.4 GetStatus exceptions

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

Table 19 — Exception codes for GetStatus operation

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

17 Upate operation (optional)

17.1 Introduction

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

17.2 Upate operation request

The following figures illustrate the UpdateRequest operation parameters in UML and XMLSpy notation.

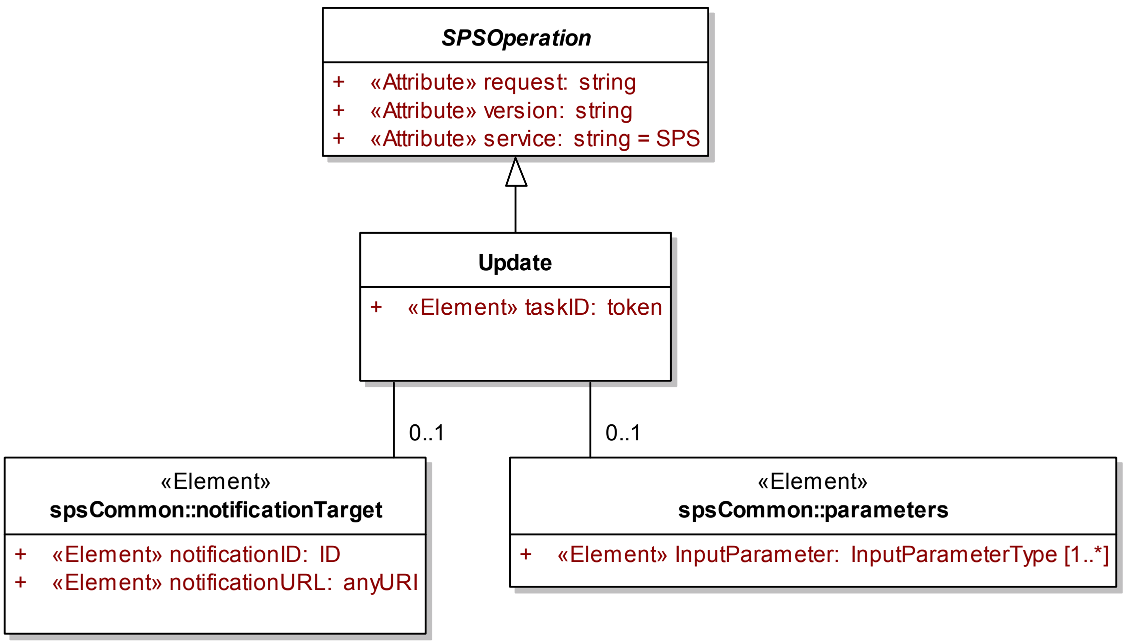


Figure 39: Upate in UML notation

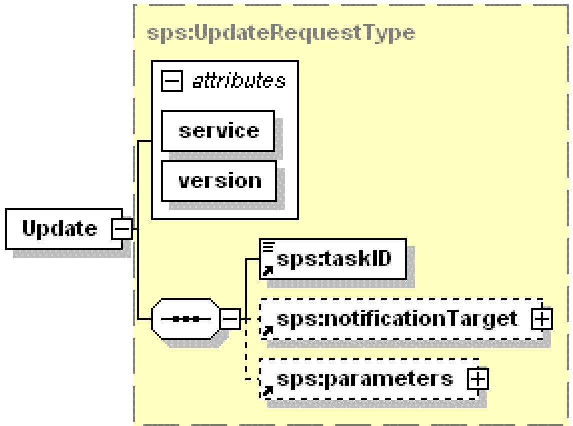


Figure 40: UpdateRequest in XMLSpy notation

17.2.1 Upate request parameters

This table also specifies the UML model data type, source of values, and multiplicity of each listed parameter, plus the meaning to servers when each optional parameter is not included in the operation request. Although some values listed in the "Name" column appear to contain spaces, they shall not contain spaces.

Table 20 — Parameters in UpdateRequest operation

Name [a]	Definition	Data type and values	Multiplicity and use
service	Service type identifier	Character String type, not empty Value is OWS type abbreviation "SPS"	One (mandatory)
request	Operation name	Character String type, not empty Value is operation name "Update"	One (mandatory)
version	Specification version for operation	Character String type, not empty Equals the number of this document	One (mandatory)
taskID	Identifier for the task that shall be updated	token	one (mandatory)
notification Target	Defines WNS data	complex, see subclause 11.2.3	one (optional)
parameters	new parameterization for the task	complex, see subclause 11.2.2	one (optional)
a The name capitalization rules being used here are specified in Subclause 11.6.2 of [OGC 05-008].			

NOTE 2 The data type of many parameters is specified as "Character String type, not empty". In the XML Schema Documents specified herein, these parameters are encoded with the xsd:string type, which does NOT require that these strings not be empty.

The "Multiplicity and use" columns in all tables specify the optionality of each listed parameter and data structure in the UpdateRequest operation. All the "mandatory" parameters and data structures shall be implemented by all SPS clients, using a specified value(s). Similarly, all the "mandatory" parameters and data structures shall be implemented by all SPS servers, checking that each request parameter or data structure is received with any specified value(s).

All the "optional" parameters and data structures, in the UpdateRequest operation, should be implemented by all SPS clients using specified values, for each implemented process to which that parameter or data structure applies. Similarly, all the "optional" parameters and data structures shall be implemented by all SPS servers, for each implemented process to which that parameter or data structure applies.

17.2.2 Upate request KVP encoding

KVP encoding not supported.

17.2.3 Upate request XML encoding

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

See annex B.

EXAMPLE An example UpdateRequest operation request XML encoded for HTTP POST is:

```
<?xml version="1.0" encoding="UTF-8"?>
<Update xmlns="http://www.opengis.net/sps"
xmlns:swe="http://www.opengis.net/swe/0" service="SPS" version="1.0.0">
  <taskID>433</taskID>
  <parameters>
    <InputParameter parameterID="pan">
      <value>
        <swe:Category>30</swe:Category>
      </value>
    </InputParameter>
    <InputParameter parameterID="tilt">
      <value>
        <swe:Category>5</swe:Category>
      </value>
    </InputParameter>
  </parameters>
</Update>
```


17.3 Upate operation response

17.3.1 Normal response parameters

The normal response to a valid UpdateRequest operation request shall be UpdateRequestResponse. More precisely, a response from the UpdateRequest operation shall include the parts listed in Table 21 and shown in the following figures. This table also specifies the UML model data type plus the multiplicity and use of each listed part.

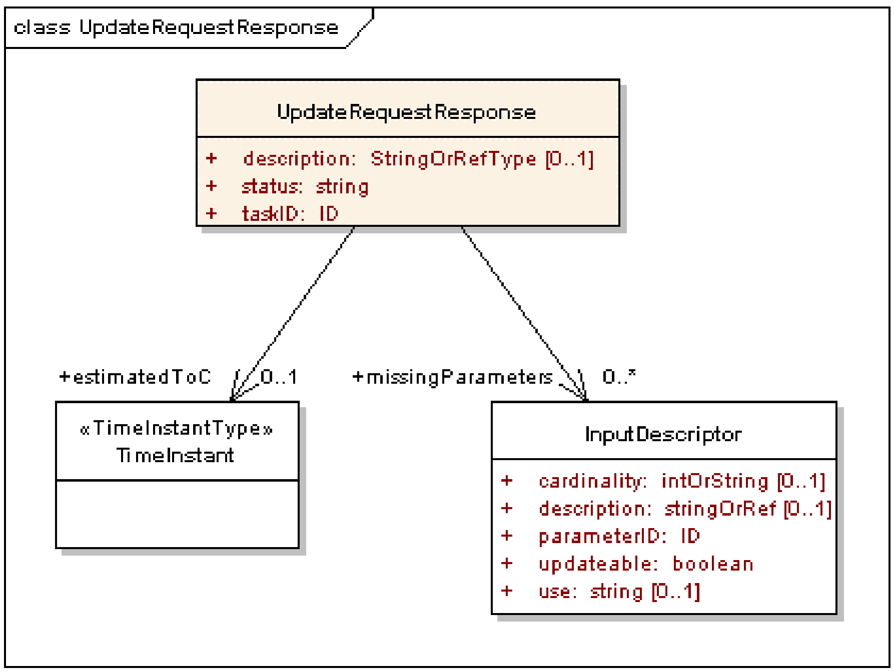


Figure 41: UpdateRequestResponse in UML notation

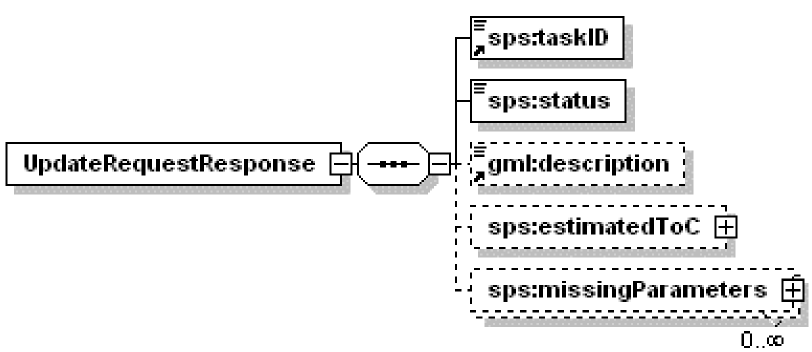


Figure 42: UpdateRequestResponse in XMLSpy notation

Table 21: — Parts of UpdateRequest operation response

Name	Definition	Data type and values	Multiplicity and use
taskID	Identifies the updated task, provided by SPS server	token	one (mandatory)
status	Tells whether the UpdateRequest has been accepted or rejected by the SPS server	String, enumerates: “confirmed”, “rejected”, “incomplete request”	one (mandatory)
description	additional continuous text	String	one (optional)
estimatedToC	Estimated Time of Completion for this task	gml:TimeInstant	one (optional)
missingParameters	Definition of additional parameters that are necessary to update a task and have not been delivered in the UpdateRequest	InputDescriptor, see subclause 11.2.2	one to many (optional)

17.3.2 Normal response XML encoding

The following schema fragment specifies the contents and structure of a Update operation response, always encoded in XML.

See annex B.

17.3.3 Upate response example

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

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

17.3.4 Upate exceptions

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

Table 22 — Exception codes for UpdateRequest operation

exceptionCode value	Meaning of code	“locator” value

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

18 Cancel operation (optional)

18.1 Introduction

The Cancel operation cancels a previously requested task.

18.2 Cancel operation request

The Cancel UML model and its representation in XMLSpy are shown in the following figures.

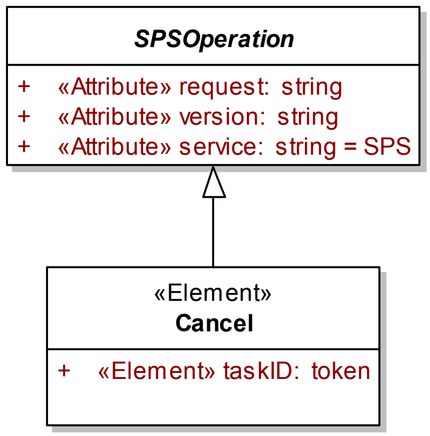


Figure 43: Cancel in UML notation

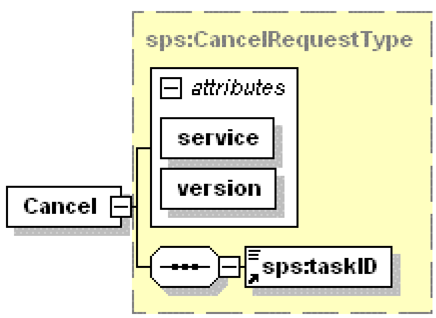


Figure 44: Cancel in XMLSpy notation

18.2.1 Cancel request parameters

This table also specifies the UML model data type, source of values, and multiplicity of each listed parameter, plus the meaning to servers when each optional parameter is not included in the operation request. Although some values listed in the "Name" column appear to contain spaces, they shall not contain spaces.

Table 23 — Parameters in Cancel operation request

Name [a]	Definition	Data type and values	Multiplicity and use
service	Service type identifier	Character String type, not empty Value is OWS type abbreviation "SPS"	One (mandatory)
request	Operation name	Character String type, not empty Value is operation name "Cancel"	One (mandatory)
version	Specification version for operation	Character String type, not empty Equals the number of this document	One (mandatory)
taskID	Identifies the task to be cancelled	token	one (mandatory)

a The name capitalization rules being used here are specified in Subclause 11.6.2 of [OGC 05-008].

NOTE 2 The data type of many parameters is specified as "Character String type, not empty". In the XML Schema Documents specified herein, these parameters are encoded with the xsd:string type, which does NOT require that these strings not be empty.

The "Multiplicity and use" columns in all tables specify the optionality of each listed parameter and data structure in the Cancel operation request. Since all parameters and data structures are mandatory in the operation request, all parameters and data structures shall be implemented by all SPS clients, using a specified value(s). Similarly, all parameters and data structures shall be implemented by all SPS servers, checking that each request parameter is received with any specified value(s).

18.2.2 Cancel request KVP encoding

KVP encoding not supported.

18.2.3 Cancel request XML encoding

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

See annex B.

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

```
<?xml version="1.0" encoding="UTF-8"?>
<Cancel xmlns="http://www.opengis.net/sps" service="SPS"
version="1.0.0">
   <taskID>433</taskID>
</Cancel>
```


18.3 Cancel operation response

18.3.1 Normal response parameters

The normal response to a valid Cancel operation request shall be a CancelRequestResponse. More precisely, a response from the Cancel operation shall include the parts listed in the following table and shown in the following figures. The table also specifies the UML model data type plus the multiplicity and use of each listed part.

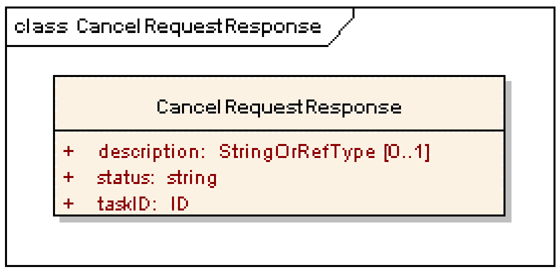


Figure 45: CancelRequestResponse in UML notation

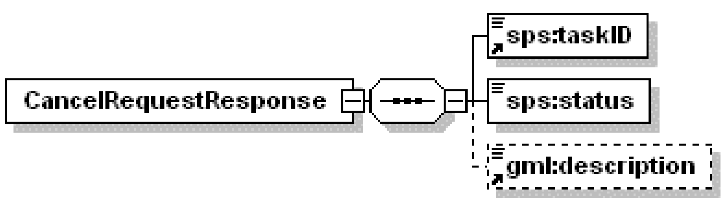


Figure 46: CancelRequestResponse in XMLSpy notation

Table 24 — Parts of Cancel operation response

Name	Definition	Data type and values	Multiplicity and use
taskID	Identifies the task	token	one (mandatory)
status	Tells whether the Cancel request has been accepted by the SPS server	String, enumerates: “confirmed”, “rejected”,	one (mandatory)
description	further information in continuous text	String	one (optional)

18.3.2 Normal response XML encoding

The following schema fragment specifies the contents and structure of a Cancel operation response, always encoded in XML.

See annex B.

18.3.3 Cancel response example

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

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

18.3.4 Cancel exceptions

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

Table 25 — Exception codes for Cancel operation

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

19 DescribeResultAccess operation (mandatory)

19.1 Introduction

The DescribeResultAccess operation allows SPS clients to retrieve information where the observed data can be accessed from. This access source may be a SOS, WMS, WFS or any other OGC Web Service that provides data.

19.2 DescribeResultAccess operation

The following figures show the DescribeResultAccess model.

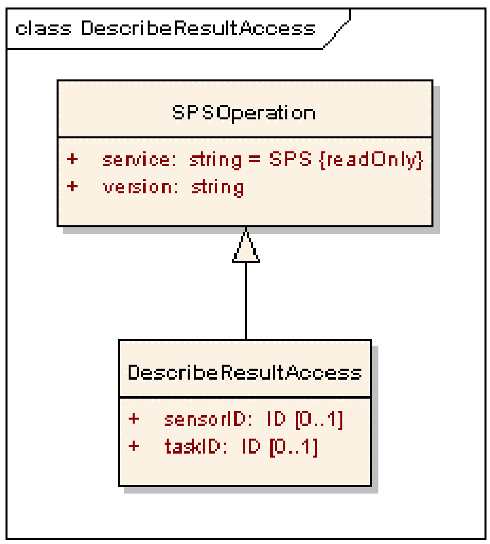


Figure 47: DescribeResultAccess request in UML notation

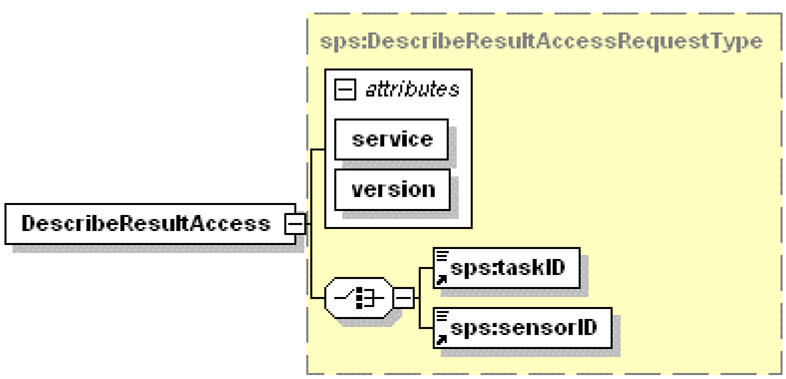


Figure 48: DescribeResultAccess in XMLSpy notation

19.2.1 DescribeResultAccess request parameters

This table also specifies the UML model data type, source of values, and multiplicity of each listed parameter, plus the meaning to servers when each optional parameter is not included in the operation request. Although some values listed in the “Name” column appear to contain spaces, they shall not contain spaces.

Table 26 — Parameters in DescribeResultAccess operation request

Name [a]	Definition	Data type and values	Multiplicity and use
service	Service type identifier	Character String type, not empty Value is OWS type abbreviation “SPS”	One (mandatory)
request	Operation name	Character String type, not empty Value is operation name ”DescribeResultAccess”	One (mandatory)
version	Specification version for operation	Character String type, not empty Equals the number of this document	One (mandatory)
taskID	Identifies task	Token	one (mandatory; if sensorID is not used)
sensorID	Identifies sensor of this task	Token	one (mandatory; if taskId is not used)
a The name capitalization rules being used here are specified in Subclause 11.6.2 of [OGC 05-008].			

NOTE 2 The data type of many parameters is specified as “Character String type, not empty”. In the XML Schema Documents specified herein, these parameters are encoded with the xsd:string type, which does NOT require that these strings not be empty.

The “Multiplicity and use” columns in all tables specify the optionality of each listed parameter and data structure in the DescribeResultAccess operation request. Since all parameters and data structures are mandatory in the operation request, all parameters and data structures shall be implemented by all SPS clients, using a specified value(s). Similarly, all parameters and data structures shall be implemented by all SPS servers, checking that each request parameter is received with any specified value(s).

19.2.2 DescribeResultAccess request KVP encoding

KVP encoding not supported.

19.2.3 DescribeResultAccess request XML encoding

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

See annex B.

EXAMPLE An example DescribeResultAccess operation request XML encoded for HTTP POST is: See annex D.

19.3 DescribeResultAccess operation response

19.3.1 Normal response parameters

The normal response to a valid DescribeResultAccess operation request shall be DescribeResultAccessRequestResponse. More precisely, a response from the DescribeResultAccess operation shall include the parts listed in the following table and shown in the following figures. This table also specifies the UML model data type plus the multiplicity and use of each listed part.

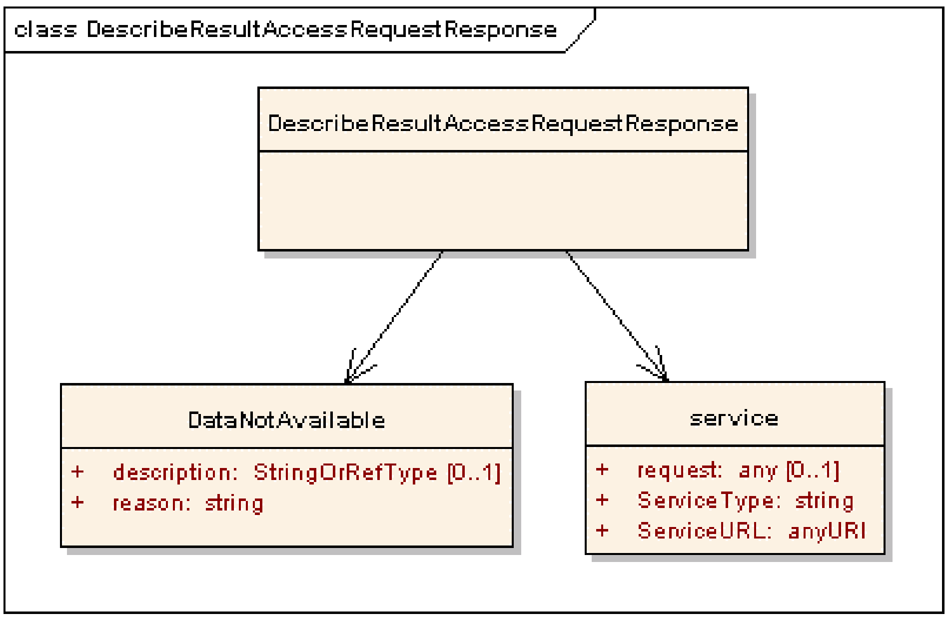


Figure 49: DescribeResultAccessRequestResponse in UML notation (Note: add an optional “request” element of type anyType to the service element.)

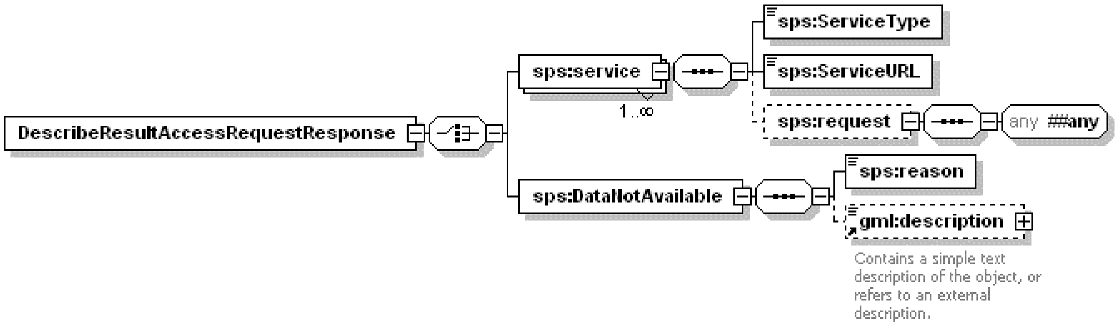


Figure 50: DescribeResultAccessRequestResponse in XMLSpy notation

Table 27 — Parts of DescribeResultAccess operation response

Name	Definition	Data type and values	Multiplicity and use
service	Element containing the type and URL of the OGC Web service that provides access to the observed data	complex, see Table 28.	one to many (mandatory)
a			

Table 28: Parts of the "service"

Name	Definition	Data type and values	Multiplicity and use
ServiceType	Defines the type of the OGC Web Service	String, example: SOS	one (mandatory)
ServiceURL	Defines the URL of the OGC Web service that provides access to the observed data	String	one (mandatory)
request	Defines the request that the client may use to retrieve results of the collection	anyType	one (optional, but: support of this element is required if the URL does not provide unambiguous access to the result dataset.)
a			

19.3.2 Normal response XML encoding

The following schema fragment specifies the contents and structure of a DescribeResultAccess operation response, always encoded in XML.

See annex B.

19.3.3 DescribeResultAccess response example

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

```
See annex D.
```


19.3.4 DescribeResultAccess exceptions

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

Table 29 — Exception codes for DescribeResultAccess operation

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

20 SPS – running example

For a better understanding we will develop a running example. This example shows how a user controls a camera using a SPS.

Imagine a user who wants to get an overview of the current situation in a specific building. This building is what we call the *area of interest* (AOI). The first step would be to call up a registry to provide descriptions of all sensors that have an area of service (AOS) which overlaps with the AOI.

The registry will return a list of SensorML descriptions for the sensors found. In our example, one of the descriptions provides information about a video camera located inside the building.

Listing 20-1: Extract of the SensorML description for video camera

```
<?xml version="1.0"?>
<SensorML xmlns="http://www.opengis.net/sensorML"
xmlns:swe="http://www.opengis.net/swe"
xmlns:xlink="http://www.w3.org/1999/xlink" version="1.0">
   <Sensor id="AXIS213_1">
      <!--~~~~~~~~~~~~~~~~~~~-->

      <!--~~~~~~~~~~~~~~~~~~~-->
      <identification>
         <IdentifierList>
            <identifier name="longName">
               <Term>IFGI Network Video Camera</Term>
            </identifier>

               <Term>ifgicam01</Term>
            </identifier>
         </IdentifierList>
      </identification>
      <classification>
         <ClassifierList>

               <Term qualifier="urn:ogc:classifier:sensorType">video
camera</Term>
            </classifier>
            <classifier name="application">
               <Term
qualifier="urn:ogc:classifier:application">surveillance and remote
monitoring for scientific purposes</Term>
            </classifier>
         </ClassifierList>
      </classification>
      <capabilities>
         <PropertyList>
            <property name="zoomRange">
               <swe:QuantityRange
definition="urn:ogc:def:phenomenon:OGC:1.0.30:FocalLength"
uom="urn:ogc:def:uom:OGC:1.0.30:mm">3.5 91</swe:QuantityRange>
            </property>
```



```xml
        <property name="resolution">
          <swe:DataGroup>
            <swe:component name="horizontalResolution">
              <swe:Count>704</swe:Count>
            </swe:component>
            <swe:component name="verticalResolution">
              <swe:Count>576</swe:Count>
            </swe:component>
          </swe:DataGroup>
        </property>
      </PropertyList>
    </capabilities>
    <!-- ... -->
    <!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
    <!--Sensor taskable parameters-->
    <!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
    <parameters>
      <ParameterList>
        <parameter name="zoom">
          <swe:Count min="1" max="9999"/>
        </parameter>
        <parameter name="pan">
          <swe:Quantity min="-180.0" max="180.0"/>
        </parameter>
        <parameter name="tilt">
          <swe:Quantity min="-180.0" max="180.0"/>
        </parameter>
        <parameter name="rzoom">
          <swe:Count min="-9999" max="9999"/>
        </parameter>
        <parameter name="rpan">
          <swe:Quantity min="-360.0" max="360.0"/>
        </parameter>
        <parameter name="rtilt">
          <swe:Quantity min="-360.0" max="360.0"/>
        </parameter>
        <parameter name="speed">
          <swe:Count min="1" max="100"/>
        </parameter>
        <parameter name="gotoserverpresetname">
          <swe:DataGroup>
            <swe:component name="allowed_value_1">
              <swe:Category>floor1</swe:Category>
            </swe:component>
            <swe:component name="allowed_value_2">
              <swe:Category>floor2</swe:Category>
            </swe:component>
          </swe:DataGroup>
        </parameter>
        <parameter name="task-start-time">
          <swe:Time/>
        </parameter>
        <parameter name="task-end-time">
          <swe:Time/>
        </parameter>
      </ParameterList>
    </parameters>
```

```
    <!-- ... -->
  </Sensor>
</SensorML>
```

Note that this description allows users to derive information about type, capabilities and taskable parameters of the sensor. The service that controls the sensor turns out to be a SPS.

Step 2: Now the user wants to find out more about the service. Note that a single SPS instance might be a façade to hundreds of sensors with even more taskable parameters. Additionally, a SPS instance might implement all or only the mandatory operations. The user sends a GetCapabilities request to the SPS instance. The response shows that the service supports all SPS operations and offers only one taskable sensor.

Step 3: Before the user moves forward in tasking the sensor he wants to find out whether he can access the sensor data. Note: The SPS is an interface **to task** an asset or asset system. It **is not** an interface to access the observational data produced by it. Observational data can be made accessible by a number of services. In most cases it might be a Sensor Observation Service, but Geo Video Service, TML Data Streaming Service, Web Feature Service, Web Coverage Service, or Web Map Service are other options.

To learn about the data access mechanism, the user fires a DescribeResultAccess request operation.

Listing 20-2: DescribeResultAccess request

```
<?xml version="1.0" encoding="UTF-8"?>
<DescribeResultAccess xmlns="http://www.opengis.net/sps" service="SPS"
version="1.0.0">
  <sensorID>
urn:x-ogc:object:sensor:IFGI:AXISPTZ_C:1:ifgicam01</sensorID>
</DescribeResultAccess>
```

The response for the DescribeResultAccess request shows that the only service that provides data from the camera is a GeoVideoService (GVS).

Listing 20-3: DescribeResultAccess response

```
<?xml version="1.0" encoding="UTF-8"?>
<DescribeResultAccessRequestResponse
xmlns="http://www.opengis.net/sps">
  <service>
    <ServiceType>TML</ServiceType>
    <ServiceURL>http://mars.uni-muenster.de:5525</ServiceURL>
  </service>
</DescribeResultAccessRequestResponse>
```


The user is satisfied with the result as he does not have a client that could access other data services than GVS, e.g. a SOS or TML-Server.

General information:

With the DescribeResultAccess operation a client can find out from which service(s) the data from a sensor offered by a SPS can be retrieved. As described previously, users use the SensorID or the taskID to identify for which sensor the access should be described.
The result contains one to many service-elements each with a service URL and a service type. The semantic of these elements for requests and responses are as follows:

DescribeResultAccess request	DescribeResultAccessRequestResponse
with sensorID	List with types of services the client can expect to get data from together with the URLs that identify which services will most probably store the data.
with taskID	Same as with sensorID but some of the service-types may be missing and URLs might have changed - may be used to ask for the services that have been determined to serve the data for a certain task.

The differentiation into these two cases should mostly not matter as for simple use cases the result would be the same. In fact, one can presume that in most of the cases, the data produced by a sensor will be available at a single service instance only. But think of following example: it might be the case that a SPS provider has a default set of data service instances that serve all the data gathered by the sensors under his control. Maybe all of these services can serve data of each sensor but the decision on which service serves the data gathered in a task is depending on the task itself. If the task produces only a short amount of data then all services might provide it but if a large amount of data is produced then maybe the service which resides on a server for heavy payloads, high traffic rates and a much bigger amount of memory will be chosen.

To conclude, you cannot make sure where data will be accessible from. Firing a request that contains a sensorID you will get the types and URLs of the services that will most likely serve the data. If you use the ID of your task you will get the information where exactly your data will be stored.

If data is being or has already been gathered in a task and becomes available at a data service the SPS shall send a notification that informs the client that he can access the data gathered in his task. The client will then have to issue a DescribeResultAccess request with taskID. The response to such a request should always return a list of those services where the data for the referenced task can be retrieved at that moment. If additional information to retrieve the data of the task is needed this information shall be included in the response, e.g. as part of the ServiceURL element. In case that no data is available yet, the response of the DescribeResultAccess operation should return a list of services designated to serve the data of that task.

Step 4: Now that the user has decided that the sensor offered could fulfill his needs it is time to task the sensor. The user issues a DescribeTasking request to the SPS to find out more about the taskable parameters of the sensor. Note: The DescribeTasking operation is important as each SPS instance may façade a different type of camera. One camera allows you to control its zoom level only, other more advanced ones even allow to task orientation, shutter speed, sensibility (visual light or infrared) or other parameters.

Listing 20-4: DescribeTasking request

```
<?xml version="1.0" encoding="UTF-8"?>
<DescribeTasking xmlns="http://www.opengis.net/sps" service="SPS"
version="1.0.0">
   <sensorID>
urn:x-ogc:object:sensor:IFGI:AXISPTZ_C:1:ifgicam01</sensorID>
</DescribeTasking>
```

The response indicates which parameters are mandatory and updatable, the required parameter format and possible constraints for each parameter. This is all information you need to send a tasking request that can be processed automatically by the SPS.

Listing 20-5: DescribeTaskingRequestResponse

```
<?xml version="1.0" encoding="UTF-8"?>
<DescribeTaskingRequestResponse
xsi:schemaLocation="http://www.opengis.net/sps
../spsAll.xsd" xmlns="http://www.opengis.net/sps"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:swe="http://www.opengis.net/swe/0"
xmlns:gml="http://www.opengis.net/gml">
  <taskingDescriptor>
     <sensorID>urn:x-
ogc:object:sensor:IFGI:AXISPTZ_C:1:ifgicam01</sensorID>
     <InputDescriptor parameterID="task-start-time"
updateable="false" use="required">
        <gml:description>Give the start-time of your task
as one dateTime-instance encoded as ISO8601. Must be before
or equal to the task-end-time.</gml:description>
        <definition>
           <commonData>
              <swe:Time definition="urn:x-
ogc:def:parameter:OGC:task-start-time">
                 <swe:uom code="urn:x-
ogc:def:uom:OGC:iso8601"/>
              </swe:Time>
           </commonData>
```



```xml
        </definition>
      </InputDescriptor>
      <InputDescriptor parameterID="task-end-time"
updateable="false" use="required">
        <gml:description>Give the end-time of your task as
one dateTime-instance encoded as ISO8601. Must be after or
equal to the task-start-time.</gml:description>
        <definition>
          <commonData>
            <swe:Time definition="urn:x-
ogc:def:parameter:OGC:task-end-time">
              <swe:uom code="urn:x-
ogc:def:uom:OGC:iso8601"/>
            </swe:Time>
          </commonData>
        </definition>
      </InputDescriptor>
      <InputDescriptor parameterID="zoom" updateable="true"
use="optional">
        <gml:description>Zooms the device n
steps.</gml:description>
        <definition>
          <commonData>
            <swe:Count definition="urn:x-
ogc:def:parameter:IFGI:Camera:AbsoluteZoom">
              <swe:uom code="urn:x-
ogc:def:uom:IFGI:Camera:Steps"/>
              <swe:constraint>
                <swe:AllowedValues>
                  <swe:min>1</swe:min>
                  <swe:max>9999</swe:max>
                </swe:AllowedValues>
              </swe:constraint>
            </swe:Count>
          </commonData>
        </definition>
      </InputDescriptor>
      <InputDescriptor parameterID="pan" updateable="true"
use="optional">
        <gml:description>Pans the device relative to the
(0,0) position.</gml:description>
        <definition>
          <commonData>
            <swe:Quantity definition="urn:x-
ogc:def:parameter:IFGI:Camera:AbsolutePan">
              <swe:uom code="urn:x-
ogc:def:uom:OGC:degree"/>
```



```
                  <swe:constraint>
                     <swe:AllowedValues>
                        <swe:min>-169</swe:min>
                        <swe:max>169</swe:max>
                     </swe:AllowedValues>
                  </swe:constraint>
               </swe:Quantity>
            </commonData>
         </definition>
      </InputDescriptor>
      <InputDescriptor parameterID="tilt" updateable="true"
use="optional">
         <gml:description>Tilts the device relative to the
(0,0) position.</gml:description>
         <definition>
            <commonData>
               <swe:Quantity definition="urn:x-
ogc:def:parameter:IFGI:Camera:AbsoluteTilt">
                  <swe:uom code="urn:x-
ogc:def:uom:OGC:degree"/>
                  <swe:constraint>
                     <swe:AllowedValues>
                        <swe:min>-90</swe:min>
                        <swe:max>10</swe:max>
                     </swe:AllowedValues>
                  </swe:constraint>
               </swe:Quantity>
            </commonData>
         </definition>
      </InputDescriptor>
      <InputDescriptor parameterID="rzoom" updateable="true"
use="optional">
         <gml:description>Zooms the device n steps relative
to the current position. Positive values mean zoom in,
negative values mean zoom out.</gml:description>
         <definition>
            <commonData>
               <swe:Count definition="urn:x-
ogc:def:parameter:IFGI:Camera:RelativeZoom">
                  <swe:uom code="urn:x-
ogc:def:uom:IFGI:Camera:Steps"/>
                  <swe:constraint>
                     <swe:AllowedValues>
                        <swe:min>-9999</swe:min>
                        <swe:max>9999</swe:max>
                     </swe:AllowedValues>
                  </swe:constraint>
               </swe:Count>
```



```
            </commonData>
         </definition>
      </InputDescriptor>
      <InputDescriptor parameterID="rpan" updateable="true"
use="optional">
         <gml:description>Pans the device n degrees relative
to the current position.</gml:description>
         <definition>
            <commonData>
               <swe:Quantity definition="urn:x-
ogc:def:parameter:IFGI:Camera:RelativePan">
                  <swe:uom code="urn:x-
ogc:def:uom:OGC:degree"/>
                  <swe:constraint>
                     <swe:AllowedValues>
                        <swe:min>-360</swe:min>
                        <swe:max>360</swe:max>
                     </swe:AllowedValues>
                  </swe:constraint>
               </swe:Quantity>
            </commonData>
         </definition>
      </InputDescriptor>
      <InputDescriptor parameterID="rtilt" updateable="true"
use="optional">
         <gml:description>Tilts the device n degrees
relative to the current position.</gml:description>
         <definition>
            <commonData>
               <swe:Quantity definition="urn:x-
ogc:def:parameter:IFGI:Camera:RelativeTilt">
                  <swe:uom code="urn:x-
ogc:def:uom:OGC:degree"/>
                  <swe:constraint>
                     <swe:AllowedValues>
                        <swe:min>-180</swe:min>
                        <swe:max>180</swe:max>
                     </swe:AllowedValues>
                  </swe:constraint>
               </swe:Quantity>
            </commonData>
         </definition>
      </InputDescriptor>
      <InputDescriptor parameterID="speed" updateable="true"
use="optional">
```

```xml
        <gml:description>Sets the head speed of the device
that is connected to the specified
camera.</gml:description>
        <definition>
           <commonData>
              <swe:Count definition="urn:x-
ogc:def:parameter:IFGI:Camera:HeadSpeed">
                 <swe:uom code="urn:x-
ogc:def:uom:IFGI:Camera:Steps"/>
                 <swe:constraint>
                    <swe:AllowedValues>
                       <swe:min>1</swe:min>
                       <swe:max>100</swe:max>
                    </swe:AllowedValues>
                 </swe:constraint>
              </swe:Count>
           </commonData>
        </definition>
     </InputDescriptor>
     <InputDescriptor parameterID="gotoserverpresetname"
updateable="true" use="optional">
        <gml:description>Move to the position associated
with one of the given values.</gml:description>
        <definition>
           <commonData>
              <swe:Category definition="urn:x-
ogc:def:parameter:IFGI:Camera:GotoPresetPosition">
                 <swe:constraint>
                    <swe:AllowedTokens>

<swe:tokenList>floor1</swe:tokenList>

<swe:tokenList>floor2</swe:tokenList>
                    </swe:AllowedTokens>
                 </swe:constraint>
              </swe:Category>
           </commonData>
        </definition>
     </InputDescriptor>
  </taskingDescriptor>
</DescribeTaskingRequestResponse>
```

Now the user knows that for submitting a request he has to provide Time-encoded values for the parameters ‘task-start-time’ and ‘task-end-time’ and is not allowed to update these parameters. He also knows that the other parameters are all optional and can be updated. If the parameter ‘speed’ is used then only values of type swe:Count (i.e. integers) between 1 and 100 are allowed. The parameter ‘gotoserverpresetname’ requires the user to use only ‘floor1’ or ‘floor2’ as value.

General information:

The DescribeTaskingRequestResponse contains information about which parameters are required and updatable. In addition the format, encoding and possible constraints of parameter values are declared.

The semantics for the ‘use’ and ‘updateable’ attributes of an InputDescriptor are as follows:

	use=required	use=optional
updateable=true	The parameter must be delivered in each GFR, SR and UR. Example: the service expects authorization-information to be delivered with this parameter, or maybe timeframe-information (see below).	The parameter may be delivered in each GFR, SR and UR. Example: the location where the sensor shall move or be moved to - like the heading of a camera or maybe a person in the field that is ordered to move to a certain location if the client wants this.
updateable=false	The parameter must be delivered in each GFR and SR, but must not be used in UR. Example: the service expects the client to deliver 'timeframe'-information which may not be altered in an UR, because that is not supported.	The parameter may be delivered in each GFR and SR, but must not be used in UR. Example: additional sensors a UAV should be equipped with for its mission - it might be impossible to add sensors when the UAV started its mission.

Complex conditions on when to use a parameter or not cannot be declared with these two attributes, e.g. 'you must deliver the parameter A if parameter B is greater than 50 and parameter C is before ten days from now'. Such a feature will remain for future versions of the SPS.

Step 5: Before the user submits the task he would like to know whether his request is feasible. Therefore he sends a GetFeasibility request to the SPS. Note: The GetFeasibility request is always important if the task might not be fulfilled for any reason. We have to differentiate between SPS that run simple simulation models and just need some parameters to run – those SPS might virtually serve any number of requests simultaneously. We do not need to ask if our Submit request will be feasible as it ever will. On the other hand, we have servers like the one described in this example. If another user wants our camera to look southwards at the moment we want to look north, the SPS has to make a decision. One of both tasks will not be feasible (might depend on priorities, IP address, random, etc.).

Listing 20-6: GetFeasibility request

```
<?xml version="1.0" encoding="UTF-8"?>
<GetFeasibility xmlns="http://www.opengis.net/sps"
xmlns:gml="http://www.opengis.net/gml"
xmlns:swe="http://www.opengis.net/swe/0" service="SPS" version="1.0.0">
   <notificationTarget>
      <notificationID>1234</notificationID>
      <notificationURL>http://mars.uni-
muenster.de:8080/WNS/wns</notificationURL>
   </notificationTarget>
   <sensorID> urn:x-
ogc:object:sensor:IFGI:AXISPTZ_C:1:ifgicam01</sensorID>
   <parameters>
      <InputParameter parameterID="zoom">
         <value>
            <swe:Count>1000</swe:Count>
         </value>
      </InputParameter>
      <InputParameter parameterID="pan">
         <value>
            <swe:Count>10</swe:Count>
         </value>
      </InputParameter>
      <InputParameter parameterID="tilt">
         <value>
            <swe:Count>0</swe:Count>
         </value>
      </InputParameter>
      <InputParameter parameterID="task-start-time">
         <value>
            <swe:Time>2005-10-05T16:25:00+02:00</swe:Time>
         </value>
      </InputParameter>
      <InputParameter parameterID="task-end-time">
         <value>
            <swe:Time>2005-10-05T16:55:00+02:00</swe:Time>
         </value>
      </InputParameter>
   </parameters>
   <timeFrame>
      <gml:TimeInstant>
         <gml:timePosition>2005-10-05T16:20:00</gml:timePosition>
      </gml:TimeInstant>
   </timeFrame>
</GetFeasibility>
```


The request shows that the user intends to use the sensor for half an hour. The user also wants to get an answer for his request five minutes before the beginning of the intended task. This means if the SPS cannot figure out if a possible Submit request will be feasible five minutes before the task should take place, the request is not longer valid.

He also provides information how he could be reached by using the notificationTarget element. This contains information on where notifications shall be sent to if necessary.

Note: SPS may need a long time to figure out if a specific request might be feasible. To overcome the problem that a user has to wait in front of its computer in order to receive the answer, SPS usually make use of Web Notification Services to send notifications to the user. The Web Notification Service basically forwards the notification using other than HTTP interface, e.g. SMS, automatic phone call, IM, and others.

The response from the service points out that his request is feasible and returns a feasibilityID generated by the service to reference the user's request.

Listing 20-7: GetFeasibilityRequestResponse

```
<?xml version="1.0" encoding="UTF-8"?>
<GetFeasibilityRequestResponse xmlns="http://www.opengis.net/sps"
xmlns:gml="http://www.opengis.net/gml">
   <feasibilityID>17</feasibilityID>
   <feasibility>feasible</feasibility>
   <LatestResponseTime>
      <gml:TimeInstant>
         <gml:timePosition>2005-10-05T16:10:00</gml:timePosition>
      </gml:TimeInstant>
   </LatestResponseTime>
</GetFeasibilityRequestResponse>
```

General information:

Using the GetFeasibility operation you can determine whether a request is feasible or not. This operation will mainly be used to find out which SPS out of a bunch of suitable services can perform a certain task. Suppose you have got three SPSs that all meet the requirements of your intended task and each service has its pros and cons. But you only need one service to perform the task. So instead of submitting a task you perform feasibility studies first. Maybe some of the services cannot even perform the task at all. This would facilitate your decision.

Feasibility studies can also be time-critical. Think of a very complex task that has to begin at a certain time and that you have to submit before that point in time. If you submit the task without a feasibility study then maybe it is not feasible and you receive the notification too late. So you do not have the time to submit the task at another

service and have a problem. Even if you perform the studies at several services simultaneously you might not receive the results from all of them in time. You might choose the first service that returns a positive answer but that service might be the most expensive one.
To solve this problem we introduced the timeFrame-element. If you use this element then you make clear that all results that are delivered after the given point in time are of no value to you, because you want to make your decision based on the results that have been delivered until then. You might even be forced to make your decision quickly after the time frame is closed if time is short.

This element might also speed up computing the result on the server side. Suppose a service knows exactly that the feasibility study cannot be performed in the time requested by the client. Then the service can just return a negative result and save computing time.

The GetFeasibilityRequestResponse contains the mandatory element 'LatestResponseTime'. This element indicates when a definite answer for a certain feasibility study will be available at the latest. The immediate response to the request will point out that the study cannot be answered at once and the definite answer will be sent later via a notification. If that notification has not been received until the 'LatestResponseTime' the feasibility study shall be deemed to be not feasible.

Step 6: As the request seems to be feasible the user submits his task. This can be done in two ways: either in providing the feasibilityID or sending a complete Submit request containing all InputParameters. The user decides to use the feasibilityID.

Listing 20-8: Submit request

```
<?xml version="1.0" encoding="UTF-8"?>
<Submit xmlns="http://www.opengis.net/sps"
xmlns:gml="http://www.opengis.net/gml" service="SPS" version="1.0.0">
   <notificationTarget>
      <notificationID>1234</notificationID>
      <notificationURL>http://mars.uni-
muenster.de:8080/WNS/wns</notificationURL>
   </notificationTarget>
   <feasibilityID>17</feasibilityID>
</Submit>
```

In case the service did not cache the feasibility request then it will respond with an ExceptionReport like the following:

Listing 20-9: ExceptionReport with exceptionCode FeasibilityIDExpired

```
<?xml version="1.0" encoding="UTF-8"?>
<ExceptionReport language="en" version="1.0.0"
xmlns="http://www.opengis.net/ows">
   <Exception exceptionCode="FeasibilityIDExpired"/>
</ExceptionReport>
```


However let us assume that the SPS does support caching. Then the SPS will submit the task if it is still feasible.

Listing 20-10: SubmitRequestResponse as sent by SPS

```
<?xml version="1.0" encoding="UTF-8"?>
<SubmitRequestResponse xmlns="http://www.opengis.net/sps"
xmlns:gml="http://www.opengis.net/gml">
   <taskID>433</taskID>
   <status>confirmed</status>
   <LatestResponseTime>
      <gml:TimeInstant>
         <gml:timePosition>2005-10-05T16:15:00</gml:timePosition>
      </gml:TimeInstant>
   </LatestResponseTime>
   <estimatedToC>
      <gml:TimeInstant>
         <gml:timePosition>2005-10-05T16:55:00+02:00</gml:timePosition>
      </gml:TimeInstant>
   </estimatedToC>
</SubmitRequestResponse>
```

Now the user knows that his task will be performed as requested. The (optional) estimatedToC element indicates that the task will be completed as requested.

General information:

If a SPS does not support caching of (feasible) feasibility requests then a request with feasibilityID will result in an ExceptionReport with exceptionCode FeasibilityIDExpired. Whenever this ExceptionCode occurs the client knows that the feasibilityID is unknown to the service. Whether caching feasibility requests is not supported or a feasibility request has been removed from the cache does not matter in this situation: the client has to use a full Submit request.

If the SPS has cached a GetFeasibility request that was responded to be feasible then this does not automatically make a Submit request with feasibilityID feasible. Maybe other tasks have been submitted in the meantime which block the intended task. The service thus will most probably check the feasibility again before submitting the task – however it might have cached information together with the feasibilityID to save computing time and thus speed up the feasibility study for the Submit request considerably.

The optional timeFrame element in a Submit request has the same meaning as in a GetFeasibility request: if a task has not been submitted until the given time frame ends the request becomes invalid and the client expects the task to have been rejected.

The ‘LatestResponseTime’ has been included in the SubmitRequestResponse as well. It has the same meaning as in GetFeasibilityRequestResponse.

Step 7: The user can perform three operations for the submitted task: Update, Cancel and GetStatus.

At 2005-10-05T16:25:01+02:00 the user performs the GetStatus operation.

Listing 20-11: GetStatus request as sent to SPS

```
<?xml version="1.0" encoding="UTF-8"?>
<GetStatus xmlns="http://www.opengis.net/sps" service="SPS"
version="1.0.0">
   <taskID>433</taskID>
</GetStatus>
```

He receives following response.

Listing 20-12: GetStatusRequestResponse as sent by SPS

```
<?xml version="1.0" encoding="UTF-8"?>
<GetStatusRequestResponse xmlns="http://www.opengis.net/sps">
   <taskID>433</taskID>
   <status>in operation</status>
</GetStatusRequestResponse>
```

So everything seems to be ok. The user’s task has begun and the camera should be collecting data for the user now.

General information:

As the name indicates the GetStatus operation allows a client to request the current status of a submitted task. This might be helpful in different situations, e.g. when the beginning of a task cannot be determined exactly. Suppose there is a queue of submitted tasks that are performed by the SPS one after another but require a certain time which is unpredictable beforehand (e.g. each task is performed by a team of engineers: sometimes the order that is implied by the task can be fulfilled very fast, sometimes this is more difficult).

The meaning of the various status codes in a GetStatusRequestResponse will be explained in the following:

value of status element	meaning
not yet started	The task is still waiting to be performed.
in operation	The task has begun.

finished	Work has been done.
cancelled	The task has been successfully cancelled by the user - or SPS operator if this was necessary. If the latter is the case then a description shall be provided via the description element and contain a message why the task has been cancelled by the operator.
delayed	The task has been delayed by the SPS operator. The description element should contain a message explaining the reason why the task has been delayed. If provided the estimatedToC element might give a hint on when the task will be completed.
unknown	It is not possible to determine the current status of the task. E.g. if a team has been sent out to gather data about a certain area or feature of interest and contact to the team has been lost. Then it is impossible to say whether the task has been finished or started or maybe even cancelled if it was impossible to fulfil.

Step 8: As the user's task has begun the SPS sends a notification to the user via the notification target (i.e. the WNS) the user provided in the Submit request. The notification informs the user that the data stream – in other words: the video stream – is now available. The user uses the GetResultAccess operation with taskID to determine where the data for his task is stored and can be accessed from.

The user displays the video stream and now wants to alter the cameras field of view. Therefore he issues an Update request that contains a bunch of parameters that are updateable.

Listing 20-13: Update request as sent to SPS

```
<?xml version="1.0" encoding="UTF-8"?>
<Update xmlns="http://www.opengis.net/sps"
xmlns:swe="http://www.opengis.net/swe/0" service="SPS" version="1.0.0">
  <taskID>433</taskID>
  <parameters>
    <InputParameter parameterID="pan">
      <value>
```



```
        <swe:Category>30</swe:Category>
      </value>
    </InputParameter>
    <InputParameter parameterID="tilt">
      <value>
        <swe:Category>5</swe:Category>
      </value>
    </InputParameter>
  </parameters>
</Update>
```

The response indicates that the Update is confirmed by the SPS.

Listing 20-14: UpdateRequestResponse

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

General information:

If a SPS supports the Update operation then all parameters that are updateable (indicated by the DescribeTaskingRequestResponse) and the notification target of a submitted task can be updated as long as the task has not been finished.

Updating the notification target might be useful if the actor responsible for the task at the client side changes or if the notification channel shall be changed, e.g. if the user no longer wants to receive notifications via e-mail but rather via SMS.

The semantics of a parameter update is up to the service. But parameters that are updateable and required have to be delivered in each Update request (see description of DescribeTaskingRequestResponse). If such a parameter is missing then the status of the response is 'incomplete request'. The service shall add a list of InputDescriptor elements to the response, for each missing parameter its corresponding InputDescriptor.

If the time of completion is altered by an update the service may add the estimatedToC element containing the new estimated time of completion.

Step 9: Having performed several updates the user has enough information about the current situation in the building. As he no longer needs the camera he cancels his task.

Listing 20-15: Cancel request as sent to SPS

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

```xml
<Cancel xmlns="http://www.opengis.net/sps" service="SPS"
version="1.0.0">
   <taskID>433</taskID>
</Cancel>
```

The response indicates that the task has been cancelled.

The whole interaction with the SPS has come to an end.

General information:

The Cancel operation can be performed on a submitted task at any time, even if it has not been started yet. The operation has two main purposes.

1. Using the Cancel operation can free resources that otherwise would be occupied unnecessarily.
   Suppose you cancel a task that would otherwise last for a longer time, say for another 30 minutes. Then you can submit a new task which otherwise would not be feasible (because it would have overlapped in time with the ongoing task).

2. Using the Cancel operation can save money.
   Suppose you have to pay for using a sensor. Depending on the payment model used by SPS a cancelled task might cost you less money than a task that has completely been performed.

Annex A
(normative)

Abstract test suite

A.1 General

A paragraph.

In each Implementation Specification document, Annex A shall specify the Abstract Test Suite, as specified in Clause 9 and Annex A of ISO 19105. That Clause and Annex specify the ISO/TC 211 requirements for Abstract Test Suites. Examples of Abstract Test Suites are available in an annex of most ISO 191XX documents, one of the more useful is in ISO 191TBD. Note that this guidance may be more abstract than needed in an OpenGIS® Implementation Specification.

Annex B
(normative)

XML Schema Documents

In addition to this document, this specification includes several normative XML Schema Documents. These XML Schema Documents are bundled in a zip file with the present document. After OGC acceptance of a Version 1.0.0 of this specification, these XML Schema Documents will also be posted online at the URL http://schemas.opengis.net/SPS/1.0.0. 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 SPS abilities now specified in this document use SPS specified XML Schema Documents included in the zip file with this document. These XML Schema Documents combine the XML schema fragments listed in various subclauses of this document, eliminating duplications. These XML Schema Documents roughly match the UML packages described in Annex B, and are named:

spsAll.xsd

spsCancelRequest.xsd, spsCancelRequestResponse.xsd

spsCommon.xsd, spsContents.xsd

spsDescribeResultAccessRequest.xsd, spsDescribeResultAccessRequestResponse.xsd,

spsDescribeTaskingRequest.xsd, spsDescribeTaskingRequestResponse.xsd

spsGetCapabilities.xsd

spsGetFeasibilityRequest.xsd, spsGetFeasibilityRequestResponse.xsd

spsGetStatusRequest.xsd, spsGetStatusRequestResponse.xsd

spsMessageSchema.xsd

spsSubmitRequest.xsd, spsSubmitRequestResponse.xsd

spsTaskMessageDictionary.xsd

spsUpdateRequest.xsd, spsUpdateRequestResponse.xsd

These XML Schema Documents use and build on the OWS common XML Schema Documents specified [OGC 05-008], named:

ows19115subset.xsd

owsCommon.xsd

owsDataIdentification.xsd

owsExceptionReport.xsd

owsGetCapabilities.xsd

owsOperationsMetadata.xsd

owsServiceIdentification.xsd

owsServiceProvider.xsd

All these XML Schema Documents contain documentation of the meaning of each element and attribute, and this documentation shall be considered normative as specified in Subclause 11.6.3 of [OGC 05-008].

B.1 spsAll.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v2004 rel. 2 U (http://www.xmlspy.com) by
Institut für Geoinformatik (Institut für Geoinformatik) -->
<schema targetNamespace="http://www.opengis.net/sps/0.0"
xmlns:sps="http://www.opengis.net/sps/0.0"
xmlns="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
version="0.0" xml:lang="en">
   <annotation>
      <appinfo>spsAll.xsd 2005/09/29</appinfo>
      <documentation>
         <description>This XML Schema includes and imports, directly and
indirectly, all the XML Schemas defined by the OGC Sensor Planning
Service (SPS).</description>
      </documentation>
   </annotation>
   <!-- ===============================================================
      includes and imports
   ============================================================== -->
   <include schemaLocation="spsCancelRequest.xsd"/>
   <include schemaLocation="spsCancelRequestResponse.xsd"/>
   <include schemaLocation="spsDescribeTaskingRequest.xsd"/>
   <include schemaLocation="spsDescribeTaskingRequestResponse.xsd"/>
   <include schemaLocation="spsDescribeResultAccessRequest.xsd"/>
   <include
schemaLocation="spsDescribeResultAccessRequestResponse.xsd"/>
   <include schemaLocation="spsGetCapabilities.xsd"/>
   <include schemaLocation="spsGetFeasibilityRequest.xsd"/>
   <include schemaLocation="spsGetFeasibilityRequestResponse.xsd"/>
   <include schemaLocation="spsGetStatusRequest.xsd"/>
   <include schemaLocation="spsGetStatusRequestResponse.xsd"/>
   <include schemaLocation="spsSubmitRequest.xsd"/>
   <include schemaLocation="spsSubmitRequestResponse.xsd"/>
   <include schemaLocation="spsUpdateRequest.xsd"/>
   <include schemaLocation="spsUpdateRequestResponse.xsd"/>
   <include schemaLocation="spsTaskMessageDictionary.xsd"/>
   <include schemaLocation="spsMessageSchema.xsd"/>
</schema>
```


B.2 spsCancelRequest.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:sps="http://www.opengis.net/sps/0.0"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.opengis.net/sps/0.0"
elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xs:include schemaLocation="./spsCommon.xsd"/>
   <!--Schema of the sps:CancelRequest-->
   <xs:element name="Cancel" type="sps:CancelRequestType">
      <xs:annotation>
         <xs:documentation>Comment describing your root
element</xs:documentation>
      </xs:annotation>
   </xs:element>
   <xs:complexType name="CancelRequestType">
      <xs:complexContent>
         <xs:extension base="sps:RequestBaseType">
            <xs:sequence>
               <xs:element ref="sps:taskID"/>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
</xs:schema>
```


B.3 spsCancelRequestResponse.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:gml="http://www.opengis.net/gml"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:sps="http://www.opengis.net/sps/0.0"
targetNamespace="http://www.opengis.net/sps/0.0"
elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xs:import namespace="http://www.opengis.net/gml"
schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/gml.xsd"/>
   <xs:include schemaLocation="./spsCommon.xsd"/>
   <!--Schema of the sps:CancelRequestResponse-->
   <xs:element name="CancelRequestResponse">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="sps:taskID"/>
            <xs:element name="status">
               <xs:simpleType>
                  <xs:restriction base="xs:string">
                     <xs:enumeration value="confirmed"/>
                     <xs:enumeration value="rejected"/>
                  </xs:restriction>
               </xs:simpleType>
            </xs:element>
            <xs:element ref="gml:description" minOccurs="0"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

B.4 spsCommon.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:swe="http://www.opengis.net/swe/0.0"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:sps="http://www.opengis.net/sps/0.0"
xmlns:gml="http://www.opengis.net/gml"
targetNamespace="http://www.opengis.net/sps/0.0"
elementFormDefault="qualified" attributeFormDefault="unqualified"
xml:lang="en">
   <!-- ===============================================================
       includes and imports
   =============================================================== -->
   <xs:import namespace="http://www.opengis.net/swe/0.0"
schemaLocation="../../sweCommon/0.0.0/swe.xsd"/>
   <xs:import namespace="http://www.opengis.net/gml"
schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/gml.xsd"/>
   <xs:annotation>
      <xs:appinfo>spsCommon.xsd 2005/06/29</xs:appinfo>
      <xs:documentation>
         <description>This XML Schema encodes the elements and types
that are shared by muliple SPS operations.</description>
         <copyright>Copyright (c) 2005 Institut for Geoinformatics
University of Muenster</copyright>
      </xs:documentation>
   </xs:annotation>
   <!-- ===============================================================
       elements and types
   =============================================================== -->
   <xs:complexType name="RequestBaseType">
      <xs:annotation>
         <xs:documentation>XML encoded SPS operation request base, for
all operations except Get Capabilities. In this XML encoding, no
"request" parameter is included, since the element name specifies the
specific operation. </xs:documentation>
      </xs:annotation>
      <xs:attribute name="service" type="xs:string" use="required"
fixed="SPS">
         <xs:annotation>
            <xs:documentation>Service type identifier.
</xs:documentation>
         </xs:annotation>
      </xs:attribute>
      <xs:attribute name="version" type="xs:string" use="required"
fixed="1.0.0">
         <xs:annotation>
            <xs:documentation>Specification version for SPS version and
operation.</xs:documentation>
         </xs:annotation>
      </xs:attribute>
   </xs:complexType>
   <xs:element name="sensorID" type="sps:sensorIDType"/>
   <xs:complexType name="sensorIDType">
      <xs:simpleContent>
```



```
        <xs:extension base="xs:token"/>
      </xs:simpleContent>
   </xs:complexType>
   <!---->
   <xs:element name="feasibilityID" type="xs:string">
      <xs:annotation>
         <xs:documentation>ID used by SPS to identify a specific
feasibility study. Could be used by user for later
reference.</xs:documentation>
      </xs:annotation>
   </xs:element>
   <!--InputParameter; used in GetFeasibilityRequest,SubmitRequest and
UpdateRequest-->
   <xs:element name="InputParameter" type="sps:InputParameterType"/>
   <xs:complexType name="InputParameterType">
      <xs:sequence>
         <xs:element name="value" maxOccurs="unbounded">
            <xs:complexType>
               <xs:sequence>
                  <xs:any processContents="skip"/>
               </xs:sequence>
            </xs:complexType>
         </xs:element>
      </xs:sequence>
      <xs:attribute name="parameterID" type="xs:ID" use="required"/>
   </xs:complexType>
   <!--Contains a List of InputParameters-->
   <xs:element name="parameters" type="sps:parametersType"/>
   <xs:complexType name="parametersType">
      <xs:sequence>
         <xs:element ref="sps:InputParameter" maxOccurs="unbounded"/>
      </xs:sequence>
   </xs:complexType>
   <!--ID of a specific SPS task-->
   <xs:element name="taskID" type="xs:token">
      <xs:annotation>
         <xs:documentation>identifies the task. Unique to every SPS
instance.</xs:documentation>
      </xs:annotation>
   </xs:element>
   <!--elements used to send notifications to the user-->
   <xs:element name="notificationTarget"
type="sps:notificationTargetType"/>
   <xs:complexType name="notificationTargetType">
      <xs:sequence>
         <xs:element name="notificationID" type="xs:token"/>
         <xs:element name="notificationURL" type="xs:anyURI"/>
      </xs:sequence>
   </xs:complexType>
   <!--TimeFrame-->
   <xs:element name="timeFrame">
      <xs:annotation>
         <xs:documentation>maximum point in time a request keeps being
valid. </xs:documentation>
      </xs:annotation>
      <xs:complexType>
         <xs:sequence>
```



```
            <xs:element ref="gml:TimeInstant"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <!--InputDescriptor: used in DescribeCollectionRequestResponse and
UpdateRequestResponse.-->
   <xs:element name="InputDescriptor">
      <xs:annotation>
         <xs:documentation>defines the input required to task a
sensor</xs:documentation>
      </xs:annotation>
      <xs:complexType>
         <xs:complexContent>
            <xs:extension base="sps:InputDescriptorType"/>
         </xs:complexContent>
      </xs:complexType>
   </xs:element>
   <xs:complexType name="InputDescriptorType">
      <xs:sequence>
         <xs:element ref="gml:description" minOccurs="0"/>
         <xs:element name="definition">
            <xs:annotation>
               <xs:documentation>defines the data block. See sml for
further information</xs:documentation>
            </xs:annotation>
            <xs:complexType>
               <xs:choice>
                  <!--  <xs:element ref="swe:DataDefinition"/>  -->
                  <xs:element name="commonData">
                     <xs:complexType>
                        <xs:sequence>
                           <xs:group ref="swe:AnyData"/>
                        </xs:sequence>
                     </xs:complexType>
                  </xs:element>
                  <xs:element name="TaskMessageDefinition"
type="xs:anyURI">
                     <xs:annotation>
                        <xs:documentation>links to a URI dictionary
whrere the taskMessage is defined properly.</xs:documentation>
                     </xs:annotation>
                  </xs:element>
                  <xs:element name="GeometryDefinition">
                     <xs:annotation>
                        <xs:documentation>enumerates gml:Point,
gml:Line, gml:Polygon as possible values</xs:documentation>
                     </xs:annotation>
                     <xs:simpleType>
                        <xs:restriction base="xs:QName">
                           <xs:enumeration value="gml:Point"/>
                           <xs:enumeration value="gml:Line"/>
                           <xs:enumeration value="gml:Polygon"/>
                        </xs:restriction>
                     </xs:simpleType>
                  </xs:element>
                  <xs:element name="TemporalDefinition">
                     <xs:annotation>
```



```
                        <xs:documentation>enumerates gml:TimeInstant
and gml:TimePeriod as possible values</xs:documentation>
                     </xs:annotation>
                     <xs:simpleType>
                        <xs:restriction base="xs:QName">
                           <xs:enumeration value="gml:TimeInstant"/>
                           <xs:enumeration value="gml:TimePeriod"/>
                        </xs:restriction>
                     </xs:simpleType>
                  </xs:element>
               </xs:choice>
            </xs:complexType>
         </xs:element>
         <xs:element name="restriction" minOccurs="0">
            <xs:annotation>
               <xs:documentation>optional. Only used if the client has
to choose one or many of the provided values.</xs:documentation>
            </xs:annotation>
            <xs:complexType>
               <xs:sequence>
                  <xs:element ref="sps:InputParameter"/>
               </xs:sequence>
            </xs:complexType>
         </xs:element>
         <xs:element name="cardinality" type="sps:cardinalityType"
minOccurs="0">
            <xs:annotation>
               <xs:documentation>Defines the number of input objects
that could be provided.</xs:documentation>
            </xs:annotation>
         </xs:element>
      </xs:sequence>
      <xs:attribute name="parameterID" type="xs:ID" use="required"/>
      <xs:attribute name="use" use="required">
         <xs:simpleType>
            <xs:restriction base="xs:string">
               <xs:enumeration value="required"/>
               <xs:enumeration value="optional"/>
            </xs:restriction>
         </xs:simpleType>
      </xs:attribute>
      <xs:attribute name="updateable" type="xs:boolean" use="optional"
default="true"/>
   </xs:complexType>
   <xs:simpleType name="cardinalityType">
      <xs:union>
         <xs:simpleType>
            <xs:restriction base="xs:int">
               <xs:minExclusive value="0"/>
            </xs:restriction>
         </xs:simpleType>
         <xs:simpleType>
            <xs:restriction base="xs:string">
               <xs:enumeration value="unbounded"/>
            </xs:restriction>
         </xs:simpleType>
      </xs:union>
```



```
   </xs:simpleType>
   <xs:element name="LatestResponseTime">
      <xs:annotation>
         <xs:documentation>Denotes the point in time when the
notification that contains the definite response to a delayed operation
request will be sent. If no notification has been received until that
time the operation shall be deemed to have been rejected or
failed.</xs:documentation>
      </xs:annotation>
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="gml:TimeInstant"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:element name="estimatedToC">
      <xs:annotation>
         <xs:documentation>Estimated Time of Completion gives a hint
when a requested operation might be completed.</xs:documentation>
      </xs:annotation>
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="gml:TimeInstant"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

B.5 spsContents.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:gml="http://www.opengis.net/gml"
xmlns:sps="http://www.opengis.net/sps/0.0"
xmlns:ows="http://www.opengis.net/ows"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.opengis.net/sps/0.0"
elementFormDefault="qualified" attributeFormDefault="unqualified"
xml:lang="en">
   <xs:annotation>
      <xs:documentation>
         <description>This XML Schema encodes the Contents section of
the SPS GetCapabilities operation response.</description>
      </xs:documentation>
   </xs:annotation>
   <!-- ===============================================================
       includes and imports
    ============================================================= -->
   <xs:import namespace="http://www.opengis.net/gml"
schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/gml.xsd"/>
   <xs:import namespace="http://www.opengis.net/ows"
schemaLocation="http://schemas.opengis.net/ows/1.0.0/owsCommon.xsd"/>
   <xs:include schemaLocation="./spsCommon.xsd"/>
   <!-- ===============================================================
       elements and types
    ============================================================= -->
   <xs:element name="Contents" type="sps:SPSContentsType">
      <xs:annotation>
         <xs:documentation>A SPS supports the discovery of itself
through a registry by two different  views. A registry could identify
suitable SPSs by either searching the capabilities for a certain type
of Phenomenon (that can be sensed by at least one sensor managed by the
SPS under investigation) in a certain target-area or by searching for
sensors with a certain ID and / or certain characteristics which are
able to sense a phenomenon in a certain target-area.</xs:documentation>
      </xs:annotation>
      <!--
======================================================================
===
       CONSTRAINTS FOR CONTENTS

       1) All SensorOfferings must have different SensorIDs.
       2) All PhenomenonOfferings must have different Phenomena.
       3) Each Phenomenon referenced by a SensorOffering must be
declared in a PhenomenonOffering.
       4) Each SensorID referenced by a PhenomenonOffering must be
declared in a SensorOffering.
       5) There may not be two identical SensorIDs in the same
PhenomenonOffering.

    ====================================================================
=========-->
```



```
      <xs:unique name="sensorOfferingKey">
         <xs:selector
xpath="./sps:SensorOfferingList/sps:SensorOffering/sps:SensorID"/>
         <xs:field xpath="."/>
      </xs:unique>
      <xs:key name="phenomenonOfferingKey">
         <xs:selector
xpath="./sps:PhenomenonOfferingList/sps:PhenomenonOffering"/>
         <xs:field xpath="sps:Phenomenon"/>
      </xs:key>
      <xs:keyref name="phenomenonOfferingKeyRef"
refer="sps:phenomenonOfferingKey">
         <xs:selector
xpath="./sps:SensorOfferingList/sps:SensorOffering"/>
         <xs:field xpath="sps:Phenomenon"/>
      </xs:keyref>
      <xs:keyref name="sensorOfferingKeyRef"
refer="sps:sensorOfferingKey">
         <xs:selector
xpath="./sps:PhenomenonOfferingList/sps:PhenomenonOffering/sps:SensorID
"/>
         <xs:field xpath="."/>
      </xs:keyref>
   </xs:element>
   <xs:complexType name="SPSContentsType">
      <xs:annotation>
         <xs:documentation>A SPS supports the discovery of itself
through a registry by two different  views. A registry could identify
suitable SPSs by either searching the capabilities for a certain type
of Phenomenon (that can be sensed by at least one sensor managed by the
SPS under investigation) in a certain target-area or by searching for
sensors with a certain ID and / or certain characteristics which are
able to sense a phenomenon in a certain target-area.</xs:documentation>
      </xs:annotation>
      <xs:sequence>
         <xs:element name="SensorOfferingList">
            <xs:complexType>
               <xs:sequence>
                  <xs:element name="SensorOffering"
type="sps:SensorOfferingType" maxOccurs="unbounded">
                     <xs:annotation>
                        <xs:documentation>Contains information
necessary to discover the abilities of the sensors managed by this
SPS.</xs:documentation>
                     </xs:annotation>
                  </xs:element>
               </xs:sequence>
            </xs:complexType>
         </xs:element>
         <xs:element name="PhenomenonOfferingList">
            <xs:complexType>
               <xs:sequence>
                  <xs:element name="PhenomenonOffering"
maxOccurs="unbounded">
                     <xs:complexType>
                        <xs:sequence>
                           <xs:element name="Phenomenon"
type="xs:anyURI">
```



```
                              <xs:annotation>
                                 <xs:documentation>Links to a URI that
holds the description of the phenomenon.</xs:documentation>
                              </xs:annotation>
                           </xs:element>
                           <xs:element name="SensorID" type="xs:token"
maxOccurs="unbounded">
                              <xs:annotation>
                                 <xs:documentation>References the
sensor that is capable of sensing the specific
phenomenon.</xs:documentation>
                              </xs:annotation>
                           </xs:element>
                        </xs:sequence>
                     </xs:complexType>
                     <xs:unique name="internalSensorIdKey">
                        <xs:selector xpath="./sps:SensorID"/>
                        <xs:field xpath="."/>
                     </xs:unique>
                  </xs:element>
               </xs:sequence>
            </xs:complexType>
         </xs:element>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="AreaOfServiceType">
      <xs:annotation>
         <xs:documentation>Contains the geometry of the area that a
certain sensor is theoretically able to collect data from. As it is not
possible to declare the exact geometry of such an area at any time (at
least for mobile sensors), this geometry should be treated as a hint
for discovering sensors that can be tasked to collect data from a
certain position or area.</xs:documentation>
      </xs:annotation>
      <xs:choice>
         <xs:element ref="ows:BoundingBox"/>
         <xs:element ref="gml:pos"/>
         <xs:element ref="gml:Polygon"/>
         <xs:element ref="gml:Solid"/>
      </xs:choice>
   </xs:complexType>
   <xs:complexType name="SensorOfferingType">
      <xs:annotation>
         <xs:documentation>Contains information necessary to discover
the abilities of the sensors managed by this SPS.</xs:documentation>
      </xs:annotation>
      <xs:sequence>
         <xs:element name="AreaOfService" type="sps:AreaOfServiceType">
            <xs:annotation>
               <xs:documentation>Contains the geometry of the area that
a certain sensor is theoretically able to collect data from. As it is
not possible to declare the exact geometry of such an area at any time
(at least for mobile sensors), this geometry should be treated as a
hint for discovering sensors that can be tasked to collect data from a
certain position or area.</xs:documentation>
            </xs:annotation>
         </xs:element>
```



```
         <xs:element name="Phenomenon" type="xs:anyURI">
            <xs:annotation>
               <xs:documentation>Links to a URI that holds the
description of the phenomenon.</xs:documentation>
            </xs:annotation>
         </xs:element>
         <xs:element name="SensorDefinition" type="xs:anyURI">
            <xs:annotation>
               <xs:documentation>Links to the SensorML-document of the
associated sensor.</xs:documentation>
            </xs:annotation>
         </xs:element>
         <xs:element name="SensorID" type="xs:token"/>
      </xs:sequence>
   </xs:complexType>
</xs:schema>
```


B.6 spsDescribeResultAccessRequest.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:sps="http://www.opengis.net/sps/0.0"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.opengis.net/sps/0.0"
elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xs:include schemaLocation="./spsCommon.xsd"/>
   <!--Schema of the sps:DescribeResultAccess-->
   <xs:element name="DescribeResultAccess"
type="sps:DescribeResultAccessRequestType">
      <xs:annotation>
         <xs:documentation> </xs:documentation>
      </xs:annotation>
   </xs:element>
   <xs:complexType name="DescribeResultAccessRequestType">
      <xs:complexContent>
         <xs:extension base="sps:RequestBaseType">
            <xs:choice>
               <xs:element ref="sps:taskID"/>
               <xs:element ref="sps:sensorID"/>
            </xs:choice>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
</xs:schema>
```

B.7 spsDescribeResultAccessRequestResponse.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:sps="http://www.opengis.net/sps/0.0"
xmlns:gml="http://www.opengis.net/gml"
targetNamespace="http://www.opengis.net/sps/0.0"
elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xs:import namespace="http://www.opengis.net/gml"
schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/gml.xsd"/>
   <xs:include schemaLocation="./spsCommon.xsd"/>
   <!--Schema of the sps:DescribeResultAccessResponse-->
   <xs:element name="DescribeResultAccessRequestResponse">
      <xs:complexType>
         <xs:choice>
            <xs:element name="service" maxOccurs="unbounded">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="ServiceType" type="xs:string"/>
                     <xs:element name="ServiceURL" type="xs:anyURI"/>
                     <xs:element name="request" minOccurs="0">
                        <xs:complexType>
                           <xs:sequence>
                              <xs:any namespace="##any"/>
                           </xs:sequence>
                        </xs:complexType>
                     </xs:element>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
            <xs:element name="DataNotAvailable">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="reason">
                        <xs:simpleType>
                           <xs:restriction base="xs:string">
                              <xs:enumeration value="not yet
available"/>
                              <xs:enumeration value="data currently
unavailable"/>
                           </xs:restriction>
                        </xs:simpleType>
                     </xs:element>
                     <xs:element ref="gml:description" minOccurs="0"/>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
         </xs:choice>
      </xs:complexType>
   </xs:element>
</xs:schema>
```


B.8 spsDescribeTaskingRequest.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:sps="http://www.opengis.net/sps/0.0"
targetNamespace="http://www.opengis.net/sps/0.0"
elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xs:include schemaLocation="./spsCommon.xsd"/>
   <!--Schema of the sps:DescribeTaskingRequest-->
   <xs:element name="DescribeTasking"
type="sps:DescribeTaskingRequestType">
      <xs:annotation>
         <xs:documentation> </xs:documentation>
      </xs:annotation>
   </xs:element>
   <xs:complexType name="DescribeTaskingRequestType">
      <xs:complexContent>
         <xs:extension base="sps:RequestBaseType">
            <xs:sequence>
               <xs:element ref="sps:sensorID" maxOccurs="unbounded"/>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
</xs:schema>
```


B.9 spsDescribeTaskingRequestResponse.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:sps="http://www.opengis.net/sps/0.0"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:gml="http://www.opengis.net/gml"
targetNamespace="http://www.opengis.net/sps/0.0"
elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xs:import namespace="http://www.opengis.net/gml"
schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/gml.xsd"/>
   <xs:include schemaLocation="./spsCommon.xsd"/>
   <!--Schema of the sps:DescribeTaskingRequestResponse-->
   <xs:element name="DescribeTaskingRequestResponse"
type="sps:DescribeTaskingRequestResponseType"/>
   <xs:complexType name="DescribeTaskingRequestResponseType">
      <xs:sequence>
         <xs:element name="taskingDescriptor" maxOccurs="unbounded">
            <xs:complexType>
               <xs:sequence>
                  <xs:element ref="sps:sensorID"/>
                  <xs:element ref="gml:description" minOccurs="0"/>
                  <xs:element ref="sps:InputDescriptor"
maxOccurs="unbounded"/>
               </xs:sequence>
            </xs:complexType>
         </xs:element>
      </xs:sequence>
   </xs:complexType>
</xs:schema>
```


B.10 spsGetCapabilities.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:sps="http://www.opengis.net/sps/0.0"
xmlns:ows="http://www.opengis.net/ows"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.opengis.net/sps/0.0"
elementFormDefault="qualified" attributeFormDefault="unqualified"
xml:lang="en">
   <xs:annotation>
      <xs:appinfo>spsGetCapabilities.xsd 2005/05/11</xs:appinfo>
      <xs:documentation>
         <description>This XML Schema encodes the SPS GetCapabilities
operation request and response.</description>
         <copyright>Copyright (c) 2005 Institut for Geoinformatics
University of Muenster</copyright>
      </xs:documentation>
   </xs:annotation>
   <!-- ===============================================================
       imports
    =============================================================== -->
   <xs:import namespace="http://www.opengis.net/ows"
schemaLocation="http://schemas.opengis.net/ows/1.0.0/owsGetCapabilities
.xsd"/>
   <xs:include schemaLocation="./spsContents.xsd"/>
   <!-- ===============================================================
       elements and types
    =============================================================== -->
   <xs:element name="GetCapabilities">
      <xs:annotation>
         <xs:documentation>Request to a SPS to perform the
GetCapabilities operation. This operation allows a client to retrieve
service metadata (capabilities XML) providing metadata for the specific
SPS server. In this XML encoding, no "request" parameter is included,
since the element name specifies the specific operation.
</xs:documentation>
      </xs:annotation>
      <xs:complexType>
         <xs:complexContent>
            <xs:extension base="ows:GetCapabilitiesType">
               <xs:attribute name="service" type="ows:ServiceType"
use="required" fixed="SPS"/>
            </xs:extension>
         </xs:complexContent>
      </xs:complexType>
   </xs:element>
   <!-- =============================================================== -->
   <xs:element name="Capabilities">
      <xs:annotation>
         <xs:documentation>XML encoded SPS GetCapabilities operation
response. This document provides clients with service metadata about a
specific service instance. If the server does not implement the
updateSequence parameter, the server shall always return the complete
Capabilities document, without the updateSequence parameter. When the
server implements the updateSequence parameter and the GetCapabilities
```



```
operation request included the updateSequence parameter with the
current value, the server shall return this element with only the
"version" and "updateSequence" attributes. Otherwise, all optional
elements shall be included or not depending on the actual value of the
Sections parameter in the GetCapabilities operation request.
</xs:documentation>
      </xs:annotation>
      <xs:complexType>
         <xs:complexContent>
            <xs:extension base="ows:CapabilitiesBaseType">
               <xs:sequence>
                  <xs:element ref="sps:Contents" minOccurs="0"/>
               </xs:sequence>
            </xs:extension>
         </xs:complexContent>
      </xs:complexType>
   </xs:element>
</xs:schema>
```


B.11 spsGetFeasibilityRequest.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:sps="http://www.opengis.net/sps/0.0"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:gml="http://www.opengis.net/gml"
targetNamespace="http://www.opengis.net/sps/0.0"
elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xs:import namespace="http://www.opengis.net/gml"
schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/gml.xsd"/>
   <xs:include schemaLocation="./spsCommon.xsd"/>
   <!--Schema of the sps:GetFeasibilityRequest-->
   <xs:element name="GetFeasibility"
type="sps:GetFeasibilityRequestType"/>
   <xs:complexType name="GetFeasibilityRequestType">
      <xs:complexContent>
         <xs:extension base="sps:RequestBaseType">
            <xs:sequence>
               <xs:element ref="sps:notificationTarget"/>
               <xs:element ref="sps:sensorID"/>
               <xs:element ref="sps:parameters"/>
               <xs:element ref="sps:timeFrame" minOccurs="0"/>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
</xs:schema>
```


B.12 spsGetFeasibilityRequestResponse

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:gml="http://www.opengis.net/gml"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:sps="http://www.opengis.net/sps/0.0"
targetNamespace="http://www.opengis.net/sps/0.0"
elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xs:import namespace="http://www.opengis.net/gml"
schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/gml.xsd"/>
   <xs:include schemaLocation="./spsCommon.xsd"/>
   <!--Response to a sps:GetFeasibilityRequest-->
   <xs:element name="GetFeasibilityRequestResponse">
      <xs:annotation>
         <xs:documentation>Reponse to a
GetFeasibilityRequest</xs:documentation>
      </xs:annotation>
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="sps:feasibilityID"/>
            <xs:element name="feasibility">
               <xs:annotation>
                  <xs:documentation>describes if a request is feasible
or not</xs:documentation>
               </xs:annotation>
               <xs:simpleType>
                  <xs:restriction base="xs:string">
                     <xs:enumeration value="feasible"/>
                     <xs:enumeration value="not feasible"/>
                     <xs:enumeration value="response delayed.
Notification will be sent."/>
                     <xs:enumeration value="request incomplete"/>
                     <xs:enumeration value="not feasible, alternatives
available"/>
                  </xs:restriction>
               </xs:simpleType>
            </xs:element>
            <xs:element ref="gml:description" minOccurs="0"/>
            <xs:element ref="sps:LatestResponseTime"/>
            <xs:element name="alternative" minOccurs="0"
maxOccurs="unbounded">
               <xs:annotation>
                  <xs:documentation>in case that the feasibility study
results in not-feasible, this elemet provides some alternative
values.</xs:documentation>
               </xs:annotation>
               <xs:complexType>
                  <xs:sequence>
                     <xs:element ref="sps:InputParameter"
maxOccurs="unbounded"/>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
         </xs:sequence>
      </xs:complexType>
```

```
   </xs:element>
</xs:schema>
```

B.13 spsGetStatusRequest.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:sps="http://www.opengis.net/sps/0.0"
targetNamespace="http://www.opengis.net/sps/0.0"
elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xs:include schemaLocation="./spsCommon.xsd"/>
   <!--Schema of the sps:GetStatusRequest-->
   <xs:element name="GetStatus" type="sps:GetStatusRequestType"/>
   <xs:complexType name="GetStatusRequestType">
      <xs:complexContent>
         <xs:extension base="sps:RequestBaseType">
            <xs:sequence>
               <xs:element ref="sps:notificationTarget" minOccurs="0"/>
               <xs:element ref="sps:taskID"/>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
</xs:schema>
```

B.14 spsGetStatusRequestReponse.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:gml="http://www.opengis.net/gml"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:sps="http://www.opengis.net/sps/0.0"
targetNamespace="http://www.opengis.net/sps/0.0"
elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xs:import namespace="http://www.opengis.net/gml"
schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/gml.xsd"/>
   <xs:include schemaLocation="./spsCommon.xsd"/>
   <!--Schema of the sps:GetStatusRequestResponse-->
   <xs:element name="GetStatusRequestResponse">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="sps:taskID"/>
            <xs:element name="status">
               <xs:annotation>
                  <xs:documentation>defines if the request is being
processed, rejected or failed to process due to insuffiencent
parametrization.</xs:documentation>
               </xs:annotation>
               <xs:simpleType>
                  <xs:restriction base="xs:string">
                     <xs:enumeration value="unknown"/>
                     <xs:enumeration value="in operation"/>
                     <xs:enumeration value="finished"/>
                     <xs:enumeration value="not yet started"/>
                     <xs:enumeration value="cancelled"/>
                     <xs:enumeration value="delayed"/>
                  </xs:restriction>
               </xs:simpleType>
            </xs:element>
            <xs:element ref="gml:description" minOccurs="0"/>
            <xs:element ref="sps:estimatedToC" minOccurs="0"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
</xs:schema>
```


B.15 spsSubmitRequest.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:sps="http://www.opengis.net/sps/0.0"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:gml="http://www.opengis.net/gml"
targetNamespace="http://www.opengis.net/sps/0.0"
elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xs:import namespace="http://www.opengis.net/gml"
schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/gml.xsd"/>
   <xs:include schemaLocation="./spsCommon.xsd"/>
   <!--Schema of the sps:SubmitRequest-->
   <xs:element name="Submit" type="sps:SubmitRequestType"/>
   <xs:complexType name="SubmitRequestType">
      <xs:complexContent>
         <xs:extension base="sps:RequestBaseType">
            <xs:sequence>
               <xs:element ref="sps:notificationTarget"/>
               <xs:choice>
                  <xs:element name="sensorParam">
                     <xs:complexType>
                        <xs:sequence>
                           <xs:element ref="sps:sensorID"/>
                           <xs:element ref="sps:parameters"/>
                        </xs:sequence>
                     </xs:complexType>
                  </xs:element>
                  <xs:element ref="sps:feasibilityID"/>
               </xs:choice>
               <xs:element ref="sps:timeFrame" minOccurs="0"/>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
</xs:schema>
```

B.16 spsSubmitRequestResponse.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:gml="http://www.opengis.net/gml"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:sps="http://www.opengis.net/sps/0.0"
targetNamespace="http://www.opengis.net/sps/0.0"
elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xs:import namespace="http://www.opengis.net/gml"
schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/gml.xsd"/>
   <xs:include schemaLocation="./spsCommon.xsd"/>
   <!--Schema of the sps:SubmitRequestResponse-->
   <xs:element name="SubmitRequestResponse">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="sps:taskID"/>
            <xs:element name="status">
               <xs:annotation>
                  <xs:documentation>defines if the request is being
processed, rejected or failed to process due to insuffiencent
parametrization.</xs:documentation>
               </xs:annotation>
               <xs:simpleType>
                  <xs:restriction base="xs:string">
                     <xs:enumeration value="confirmed"/>
                     <xs:enumeration value="rejected"/>
                     <xs:enumeration value="incomplete request"/>
                     <xs:enumeration value="pending"/>
                     <xs:enumeration value="rejected, alternatives
available"/>
                  </xs:restriction>
               </xs:simpleType>
            </xs:element>
            <xs:element ref="gml:description" minOccurs="0"/>
            <xs:element ref="sps:LatestResponseTime"/>
            <xs:element ref="sps:estimatedToC" minOccurs="0"/>
            <xs:element name="alternative" minOccurs="0"
maxOccurs="unbounded">
               <xs:annotation>
                  <xs:documentation>in case a request is rejected, ths
element may provide some alternative values.</xs:documentation>
               </xs:annotation>
               <xs:complexType>
                  <xs:sequence>
                     <xs:element ref="sps:InputParameter"
maxOccurs="unbounded"/>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
</xs:schema>
```


B.17 spsUpdateRequest.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:sps="http://www.opengis.net/sps/0.0"
targetNamespace="http://www.opengis.net/sps/0.0"
elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xs:include schemaLocation="./spsCommon.xsd"/>
   <!--Schema of the sps:UpdateRequest-->
   <xs:element name="Update" type="sps:UpdateRequestType"/>
   <xs:complexType name="UpdateRequestType">
      <xs:complexContent>
         <xs:extension base="sps:RequestBaseType">
            <xs:sequence>
               <xs:element ref="sps:taskID"/>
               <xs:element ref="sps:notificationTarget" minOccurs="0"/>
               <xs:element ref="sps:parameters" minOccurs="0"/>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
</xs:schema>
```

B.18 spsUpdateRequestResponse.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:sps="http://www.opengis.net/sps/0.0"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:gml="http://www.opengis.net/gml"
targetNamespace="http://www.opengis.net/sps/0.0"
elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xs:import namespace="http://www.opengis.net/gml"
schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/gml.xsd"/>
   <xs:include schemaLocation="./spsCommon.xsd"/>
   <!--Schema of the sps:UpdateRequestResponse-->
   <xs:element name="UpdateRequestResponse">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="sps:taskID"/>
            <xs:element name="status">
               <xs:simpleType>
                  <xs:restriction base="xs:string">
                     <xs:enumeration value="confirmed"/>
                     <xs:enumeration value="rejected"/>
                     <xs:enumeration value="request incomplete"/>
                  </xs:restriction>
               </xs:simpleType>
            </xs:element>
            <xs:element ref="gml:description" minOccurs="0"/>
            <xs:element name="estimatedToC" minOccurs="0">
               <xs:annotation>
```



```
                  <xs:documentation>Estimated Time of Completion gives
a hint when a requested operation might be
completed.</xs:documentation>
               </xs:annotation>
               <xs:complexType>
                  <xs:sequence>
                     <xs:element ref="gml:TimeInstant"/>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
            <xs:element name="missingParameters" minOccurs="0"
maxOccurs="unbounded">
               <xs:annotation>
                  <xs:documentation>in case that an update request was
incomplete, missingParameters describe the kind of further input
necessary to process the required update. Necessary due to possible
indenpendencies of parameters.</xs:documentation>
               </xs:annotation>
               <xs:complexType>
                  <xs:sequence>
                     <xs:element ref="sps:InputDescriptor"/>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

B.19 spsMessageSchema.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:gml="http://www.opengis.net/gml"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="http://www.opengis.net/sps/0.0"
targetNamespace="http://www.opengis.net/sps/0.0"
elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xs:include schemaLocation="./spsCommon.xsd"/>
   <xs:import namespace="http://www.opengis.net/gml"
schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/gml.xsd"/>
   <!--Normal XML Message-->
   <xs:element name="SPSMessage">
      <xs:complexType>
         <xs:choice>
            <xs:element name="FeasibilityResponse">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="feasibility">
                        <xs:simpleType>
                           <xs:restriction base="xs:token">
                              <xs:enumeration value="feasible"/>
                              <xs:enumeration value="not feasible"/>
```



```
                              <xs:enumeration value="not feasible,
alternatives available"/>
                           </xs:restriction>
                        </xs:simpleType>
                     </xs:element>
                     <xs:element ref="gml:description" minOccurs="0"/>
                     <xs:element name="alternative" minOccurs="0">
                        <xs:complexType>
                           <xs:sequence>
                              <xs:element ref="InputParameter"
maxOccurs="unbounded"/>
                           </xs:sequence>
                        </xs:complexType>
                     </xs:element>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
            <xs:element name="SubmitResponse">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="status">
                        <xs:simpleType>
                           <xs:restriction base="xs:token">
                              <xs:enumeration value="confirmed"/>
                              <xs:enumeration value="rejected"/>
                              <xs:enumeration value="rejected,
alternatives available"/>
                           </xs:restriction>
                        </xs:simpleType>
                     </xs:element>
                     <xs:element ref="gml:description" minOccurs="0"/>
                     <xs:element ref="estimatedToC" minOccurs="0"/>
                     <xs:element name="alternative" minOccurs="0">
                        <xs:complexType>
                           <xs:sequence>
                              <xs:element ref="InputParameter"
maxOccurs="unbounded"/>
                           </xs:sequence>
                        </xs:complexType>
                     </xs:element>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
            <xs:element name="StatusInformation">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="status">
                        <xs:simpleType>
                           <xs:restriction base="xs:token">
                              <xs:enumeration value="Operation
commencing"/>

                              <xs:enumeration value="Operation
completed"/>
                              <xs:enumeration value="Operation
failed"/>
                              <xs:enumeration value="Operation
cancelled"/>
```



```
                              <xs:enumeration value="Operation
delayed"/>
                              <xs:enumeration value="Operation delayed,
update needed"/>
                              <xs:enumeration value="New data
available"/>
                              <xs:enumeration value="status unknown"/>
                           </xs:restriction>
                        </xs:simpleType>
                     </xs:element>
                     <xs:element ref="gml:description" minOccurs="0"/>
                     <xs:element ref="estimatedToC" minOccurs="0"/>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
            <xs:element name="UpdateRequest">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element ref="InputDescriptor"
maxOccurs="unbounded"/>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
            <xs:element name="UpdateResponse">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element ref="InputParameter"
maxOccurs="unbounded"/>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
         </xs:choice>
         <xs:attribute name="SPSCorrID" type="xs:token" use="required">
            <xs:annotation>
               <xs:documentation>Used to correlate an incoming message
with a certain feasibility study or task performed by the
SPS.</xs:documentation>
            </xs:annotation>
         </xs:attribute>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

B.20 spsTaskMessageDictionary.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:sps="http://www.opengis.net/sps/0.0"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:gml="http://www.opengis.net/gml"
targetNamespace="http://www.opengis.net/sps/0.0"
elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xs:import namespace="http://www.opengis.net/gml"
schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/gml.xsd"/>
```



```
   <xs:element name="TaskMessageDictionary" type="gml:DictionaryType"/>
   <xs:element name="TaskMessageDefinition"
type="sps:TaskMessageDefinitionType"
substitutionGroup="gml:Definition">
      <xs:annotation>
         <xs:documentation>Derived from gml:Definition, the
TaskMessageDefinition describes the task message</xs:documentation>
      </xs:annotation>
   </xs:element>
   <xs:complexType name="TaskMessageDefinitionType">
      <xs:complexContent>
         <xs:extension base="gml:DefinitionType">
            <xs:sequence>
               <xs:element name="externalDefinition" type="xs:anyURI"
minOccurs="0">
                  <xs:annotation>
                     <xs:documentation>in case that further information
can be found externally.</xs:documentation>
                  </xs:annotation>
               </xs:element>
               <xs:element name="type" minOccurs="0">
                  <xs:annotation>
                     <xs:documentation>differentiates alpanumerical,
symbol and numerical values.</xs:documentation>
                  </xs:annotation>
                  <xs:simpleType>
                     <xs:restriction base="xs:string">
                        <xs:enumeration value="symbol"/>
                        <xs:enumeration value="numerical"/>
                        <xs:enumeration value="alphanumeric"/>
                     </xs:restriction>
                  </xs:simpleType>
               </xs:element>
               <xs:element name="length" minOccurs="0">
                  <xs:annotation>
                     <xs:documentation>describes the number of
charakters of the value.</xs:documentation>
                  </xs:annotation>
               </xs:element>
               <xs:element name="min" minOccurs="0"/>
               <xs:element name="max" minOccurs="0"/>
               <xs:element name="structureSchema" type="xs:anyURI"
minOccurs="0">
                  <xs:annotation>
                     <xs:documentation>links to the schema describing
the structure of this task message.</xs:documentation>
                  </xs:annotation>
               </xs:element>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
</xs:schema>
```

Annex C
(informative)

UML model

C.1 Introduction

This annex provides a UML model of the SPS interface, using the OGC/ISO profile of UML summarized in Subclause 5.3 of [05-008].

Figure C.1 is a simple UML diagram summarizing the SPS interface. This class diagram shows that the SPS class inherits the getCapabilities operation from the OGCWebService interface class, and adds the SPS operations. (The capitalization of names uses the OGC/ISO profile of UML.)

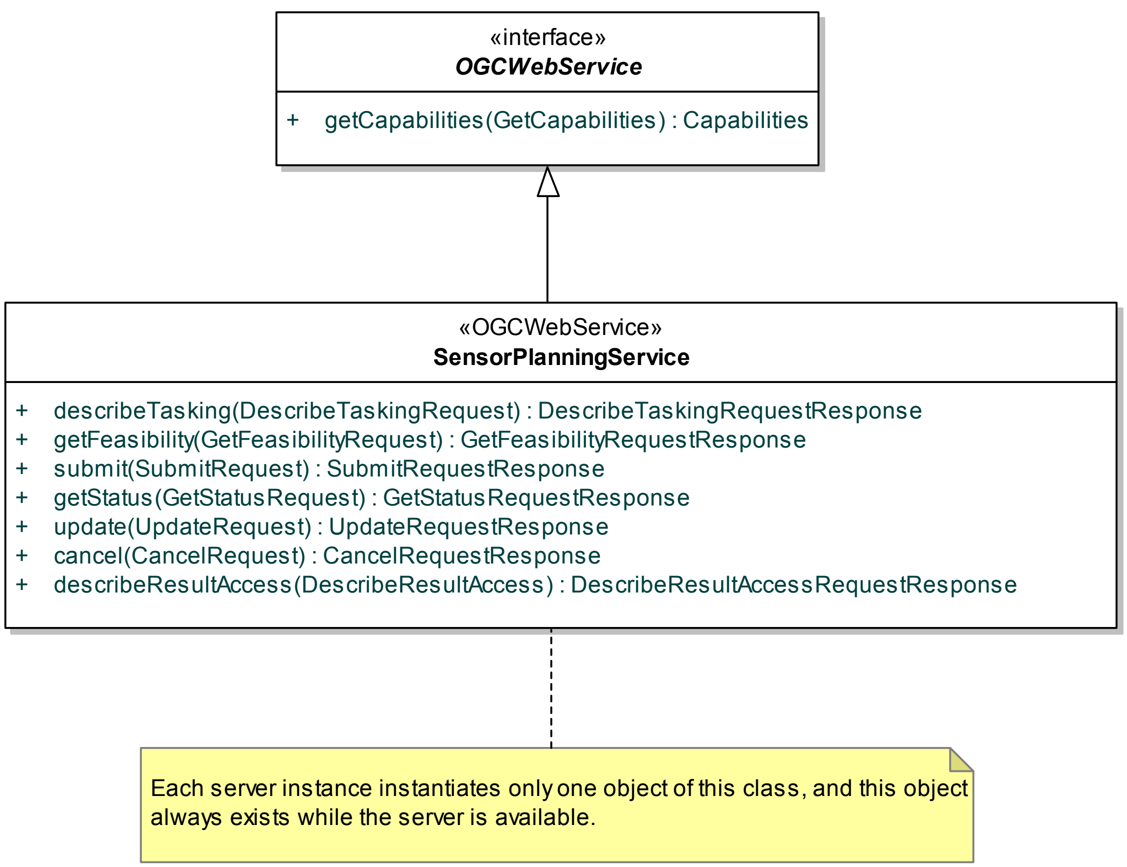


Figure C.1 — SPS interface UML diagram

Each of the SPS operations uses a request and a response data type, each of which is defined by one or more additional UML classes. The complete UML model is part of the main document and will not be repeated as part of this annex.

Annex D
(informative)

Example XML documents

D.1 Introduction

This annex provides more example XML documents than given in the body of this document.

D.2 GetCapabilitiesRequestResponse


```xml
<?xml version="1.0" encoding="UTF-8"?>
<Capabilities version="1.0.0" updateSequence=""
xsi:schemaLocation="http://www.opengis.net/sps ../spsAll.xsd"
xmlns="http://www.opengis.net/sps"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ows="http://www.opengis.net/ows"
xmlns:xlink="http://www.w3.org/1999/xlink">
    <ows:ServiceIdentification>
        <ows:Title>IFGI SPS</ows:Title>
        <ows:Abstract>Sensor planning service managing a surveillance
camera.</ows:Abstract>
        <ows:Keywords>
            <ows:Keyword>SPS</ows:Keyword>
            <ows:Keyword>network</ows:Keyword>
            <ows:Keyword>camera</ows:Keyword>
            <ows:Keyword>surveillance</ows:Keyword>
            <ows:Keyword>monitoring</ows:Keyword>
        </ows:Keywords>
        <ows:ServiceType
codeSpace="http://opengeospatial.net">OGC:SPS</ows:ServiceType>
        <ows:ServiceTypeVersion>1.0.0</ows:ServiceTypeVersion>
        <ows:Fees>NONE</ows:Fees>
        <ows:AccessConstraints>NONE</ows:AccessConstraints>
    </ows:ServiceIdentification>
    <ows:ServiceProvider>
        <ows:ProviderName>Institute for Geoinformatics, University of
Muenster</ows:ProviderName>
        <ows:ProviderSite xlink:href="http://ifgi.uni-muenster.de"/>
        <ows:ServiceContact>
            <ows:IndividualName>Johannes Echterhoff</ows:IndividualName>
            <ows:PositionName>graduate student</ows:PositionName>
            <ows:ContactInfo>
                <ows:Phone>
                    <ows:Voice>+49 251 83 39761</ows:Voice>
                    <ows:Facsimile>+49 251 83 39763</ows:Facsimile>
                </ows:Phone>
                <ows:Address>
```



```xml
                    <ows:DeliveryPoint>Institute for Geoinformatics,
University of Muenster</ows:DeliveryPoint>
                    <ows:City>Muenster</ows:City>
                    <ows:AdministrativeArea>NRW</ows:AdministrativeArea>
                    <ows:PostalCode>48149</ows:PostalCode>
                    <ows:Country>Germany</ows:Country>
                    <ows:ElectronicMailAddress>echterhoff@uni-
muenster.de</ows:ElectronicMailAddress>
                </ows:Address>
            </ows:ContactInfo>
        </ows:ServiceContact>
    </ows:ServiceProvider>
    <Contents>
        <SensorOfferingList>
            <SensorOffering>
                <AreaOfService>
                    <ows:WGS84BoundingBox>
                        <ows:LowerCorner>7.6069360670
51.9599905107</ows:LowerCorner>
                        <ows:UpperCorner>7.6075430075
51.9603508343</ows:UpperCorner>
                    </ows:WGS84BoundingBox>
                </AreaOfService>
                <Phenomenon>urn:x-
ogc:def:phenomenon:OGC:Radiance</Phenomenon>
                <SensorDefinition>http://mars.uni-
muenster.de:8080/52nPM/sml/ifgicamSML.xml</SensorDefinition>
                <SensorID>urn:x-
ogc:object:sensor:IFGI:AXISPTZ_C:1:ifgicam01</SensorID>
            </SensorOffering>
        </SensorOfferingList>
        <PhenomenonOfferingList>
            <PhenomenonOffering>
                <Phenomenon>urn:x-
ogc:def:phenomenon:OGC:Radiance</Phenomenon>
                <SensorID>urn:x-
ogc:object:sensor:IFGI:AXISPTZ_C:1:ifgicam01</SensorID>
            </PhenomenonOffering>
        </PhenomenonOfferingList>
    </Contents>
</Capabilities>
```


Annex E
(informative)

The ACTM : An Example Task Message Definition

E.1 Introduction

The material in this annex describes an example Task Message Definition, the Aircraft Collection Tasking Message (ACTM). It is intended to provide a very general general structure for tasking collections using airborne or other mobile data acquisition assets.

E.2 The ACTM Data Dictionary

Field Name	Field Definition	Field Usage *	Type**	Size	Value Range & Structure
ACTM Metadata					
Security Classification	Security classification of the Aircraft Collection Tasking Message (ACTM). Will contain a "Source" element and either "Classification String" or "Classification Any"				
Source	Source country/organization of the security classification	M	A	3	3-letter country code, or "NAT" for NATO
Classification String	Free-text classification of ACTM, either this field or "Classification Any" must be used.	C	A/N/S	128	Per source country/organizaion classification guide
Classification Any	Alternative means of classification from external xml library	C	any	N/A	The Classification tag uses the xs:any schema element, which means that this tag may contain any string or as-yet-undefined XML element, to accomodate current and future classification markings.
Sensitive Compartmented Information (SCI) Control Systems/Codewords	SCI control systems and code words.	C	A/S	35	If the ACTM contains Sensitive Compartmented Information (SCI), SCI Control Systems/Codewords must be used…field is therefore conditional.
Dissemination Control Markings	Dissemination control codewords.	C	A/S	35	If releasable, use the Authorized for Release to (REL TO) marking followed by country trigraphs in alphabetical order. Use country trigraphs from International Organization for Standardization (ISO) 3166. If dissemination of the ACTM must be further controlled, Dissemination Control Markings must be used here...field is therefore conditional.
Declassification Information	Declassification information for the ACTM.	C	A/N/S	35	If the ACTM is classified, Declassification Information must be used…field is therefore conditional.

Task Generation Timestamp	Date/time the ACTM file was created or modified. Creation Timestamp will change anytime there is a change to the ACTM.	O	A/N/S	20	"YYYY-MM-DDThh:mm:ssZ", which is a standard xs:dateTime format with a mandatory "Z" timezone specifier on the end. Task Generation Timestamp will change with any changes to the ACTM. So, for every Requirement ID control date change, there will be a Task Generation Timestamp change. But, there will not be a control date change for every Task Generation Timestamp change, ie Mission Number change with all the same requirements.
Originator	Unit which originated the requirement.	O	A/N/S	N/A	Free text. Size of field unbounded.
Mission					
Mission Number	One of two ways to Identify a particular mission, either this 10 character "Mission Number" or the 54 Character "Mission Identifier" must be used.	C	A/N	10	Alphanumeric...actually made up of the fields below: Program Code, Month, Sortie and Year.
Program Code	A theater-specific code which is part of the mission number.	C	A/N	2	Theater code (A/N) followed by A/N. See theater-specific instructions.
Month	2-digit Month, part of Mission Number	C	N	2	NN
Sortie	The sortie number associated with the project code.	C	N	2	01-99
Year	Four digit year	C	N	4	NNNN
Mission Identifier	Alternate to "Mission Number"…one of the two must be filled out in the message	C	A/N/S	54	Free-text. See theater-specific instructions.
Project Code	Theater designated code	O	A	2	Alpha, AA-ZZ. See theater-specific instructions.
Track Number	Theater designation used to identify the aircraft track (or orbit). This is the path on the earth's surface directly beneath the flight path of the airborne platform. The path may be geographically coincident to that of a scene centerline.	O	A/N	6	Alphanumeric. See theater-specific instructions.
Mission Nickname	Theater designation used to identify the aircraft track (or orbit) name.	O	A/N	N/A	Alphanumeric. Size of field unbounded.
Collection Requirements					

Requirement Identification (ID)	User specified identification to facilitate tracking collection requirements.	M	A/N/S	12	User defined
Requirement Name	User specified text name providing descriptive data about the collection requirement purpose; usually the same as the nomination name.	O	A/N	50	User defined
Requestor ID	Code for the unit which created the ACTM.	O	A/N/S	50	Free text
Control Date	Date/Time the Requirement ID was created or modified. The Requirement ID may not change if the exact same tasks were flown over several days.	M	A/N/S	20	"YYYY-MM-DDThh:mm:ssZ", which is a standard xs:dateTime format with a mandatory "Z" timezone specifer on the end. Control Date changes only when there is a change to the Requirement ID. So, for every Requirement ID control date change, there will be a Creation Timestamp change. But, there will not be a control date change for every Creation Timestamp change, ie Mission Number change with all the same requirements.
Requirement Priority	The relative importance of the collection requirement, where the lower assigned number value, the higher the priority.	M	N	3	000-999
Tasks					
Task Priority	The relative importance of the task, where the lower assigned number value, the higher the priority.	M	N	3	000-999
Essential Elements of Information (EEI) ID	Identification code for an essential element of information set within a task. EEI identification codes are defined by the user in the Exploitation Requirements section of the ACTM.	O	A/N	N/A	Alphanumeric: "EEI_NNNN"…size of field unbounded
Special Collection Instructions	Free-text explanation of any extraordinary instructions for the collecting platform. For example, a "pursue" request for FMV.	O	A/N/S	N/A	Narrative. Size of field unbounded.
Target Data					
Target Type	Indicates the collection type for the target to be collected as either a point, LOC, or DSA				
Point Target					

BE Number	The Basic Encyclopedia (BE) number of the target.	O	A/N/S	15	Possible values are: BE number (10 A/N/S) BE w/suffix (15 A/N/S) BE w/suffix (KAI) (15 A/N/S)
Target Name	The target name which generally provides information about the target's location and/or function.	M	A/N/S	38	Alphanumeric and symbol
Target Category Code	The target category code is based on a breakdown of targets into major groups.	O	N	N/A	Unbounded. See theater-specific instructions.
Center Coordinates					
Latitude	The latitude of the point target associated with the target's BE number.	M	A/N	11	DDMMSS.SSSH where DD 00-90 degrees; MM 00-59 minutes; SS.SSS 00.000-59.000 seconds; H hemisphere N (north) or S (south)
Longitude	The longitude of the point target associated with the target's BE number.	M	A/N	12	DDDMMSS.SSSH where DD 00-180 degrees; MM 00-59 minutes; SS.SSS 00.000-59.000 seconds; H hemisphere E (east) or W (west)
Elevation	The elevation of the point target associated with the target's BE number.	O	S/N	6	SNNNNN; -00680 to +10688 (in meters, mean sea level)
Major Axis	The long axis of a point target as determined by an imaginary ellipse around the target center point coordinates.	O	N	6	NNNN.N; 00.1 - 9999.9 meters
Minor Axis	The short axis of a point target as determined by an imaginary ellipse around the target center point coordinates.	O	N	6	NNNN.N; 00.1 - 9999.9 meters
True North Angle	The angle measured in degrees from True North clockwise to the major axis of the target.	O	N	3	NNN, 000-180 degrees
Look At Coordinates					
Latitude	The "look at" center point latitude when this coordinate is different from the target center latitude. This is the coordinate latitude where the requestor wants the collection focused.	C	A/N	11	DDMMSS.SSSH where DD 00-90 degrees; MM 00-59 minutes; SS.SSS 00.000-59.000 seconds; H hemisphere N (north) or S (south). When Look at Coords are used, latitude and longitude must be provided…field is therefore conditional.

Longitude	The "look at" center point longitude when this coordinate is different from the target center point longitude. This is the coordinate longitude where the requestor wants the collection focused.	C	A/N	12	DDDMMSS.SSSH where DD 00-180 degrees; MM 00-59 minutes; SS.SSS 00.000-59.000 seconds; H hemisphere E (east) or W (west). When Look at Coords are used, latitude and longitude must be provided…field is therefore conditional.
Elevation	The elevation of the point target as expressed by the "look at" (collection) coordinates.	O	S/N	6	SNNNNN; -00680 to +10688 meters mean sea level
Major Axis	The long axis of a point target as determined by an imaginary ellipse around the target center point coordinates.	O	N	6	NNNN.N; 00.1 - 9999.9 meters
Minor Axis	The short axis of a point target as determined by an imaginary ellipse around the target center point coordinates.	O	N	6	NNNN.N; 00.1 - 9999.9 meters
True North Angle	The angle measured in degrees from True North clockwise to the major axis of the target.	O	N	3	NNN, 000-180 degrees
Directed Search Area (DSA) Target					
DSA ID	The target identifier of the DSA.	O	A/N/S	15	Unrestricted
Target Name	The target name which generally provides information about the target's location and/or function.	M	A/N/S	38	Alphanumeric and symbol
Vertex Count	For a DSA, this field indicates the number of vertices in the polygon.	M	N	2	03-24
Vertex Coordinates					
Latitude	The latitude for each vertex in a DSA collection.	M	A/N	11	DDMMSS.SSSH where DD 00-90 degrees; MM 00-59 minutes; SS.SSS 00.000-59.000 seconds; H hemisphere N (north) or S (south)
Longitude	The longitude for each vertex in a DSA collection.	M	A/N	12	DDDMMSS.SSSH where DD 00-180 degrees; MM 00-59 minutes; SS.SSS 00.000-59.000 seconds; H hemisphere E (east) or W (west)
Line of Communication (LOC) Target					
LOC ID	The target identifier of the LOC.	O	A/N/S	15	Unrestricted
Target Name	The target name which generally provides information about the target's location and/or function.	M	A/N/S	38	Alphanumeric and symbol

Vertex Count	Indicates the number of vertices in the LOC.	M	N	2	02-24
Default Min Swath Width	The minimum swath width, in meters, required along the entire LOC, unless overridden by a minimum segment width.	O	N	6	000001 - 999999
Segment Vertex					
Latitude	The latitude for the segment start point in an LOC collection.	M	A/N	11	DDMMSS.SSSH where DD 00-90 degrees; MM 00-59 minutes; SS.SSS 00.000-59.000 seconds; H hemisphere N (north) or S (south)
Longitude	The longitude for the segment start point in an LOC collection.	M	A/N	12	DDDMMSS.SSSH where DD 00-180 degrees; MM 00-59 minutes; SS.SSS 00.000-59.000 seconds; H hemisphere E (east) or W (west)
Minimum Segment Swath Width	The minimum swath width, in meters, required along the segment beginning with this vertex	O	N	6	000001 - 999999
Elevation	The elevation for the segment start point in an LOC collection.	O	S/N	6	SNNNNN; -00680 to +10688 meters mean sea level
Area or Sector Blanking					
Area or Sector ID	Identification code of an area where collection is denied/not desired	O	A/N/S	N/A	Unbounded. See theater-specific instructions.
Area or Sector Name	Name of an area where collection is not permitted	M	A/N/S	N/A	Free text. Size of field unbounded.
Vertex Count	Indicates the number of vertices in the Area or Sector.	M	N	2	24-Mar
Vertex Coordinates					
Latitude	The latitude for each vertex in an area.	M	A/N	11	DDMMSS.SSSH where DD 00-90 degrees; MM 00-59 minutes; SS.SSS 00.000-59.000 seconds; H hemisphere N (north) or S (south)
Longitude	The longitude for each vertex in an area.	M	A/N	12	DDDMMSS.SSSH where DD 00-180 degrees; MM 00-59 minutes; SS.SSS 00.000-59.000 seconds; H hemisphere E (east) or W (west)
Country Code					
Country Code2	Two-letter digraph defining the country for the reference point of the image.	O	A	2	Alpha. Standard codes may be found in International Organization for Standardization (ISO)

					3166.
Country Code3	Three-letter trigraph defining the country for the reference point of the image.	O	A	3	Alpha. Standard codes may be found in International Organization for Standardization (ISO) 3166.
Geographic Region	The geographic region in which the target is located.	O	A/N	2	Alphanumeric. See theater-specific instructions.
Collection Constraints					
Sensor ID	Identifies the specific sensor which will produce the image.	M	A/N	6	Alphanumeric designations derived from STANAG 4545 and STDI-002
Sensor Mode	Identifies the mode in which the sensor will operate.	O	N	3	001-999, based on STANAG 4545 and STDI-002.
Sensor Specific Data					
Optical					
Imagery Interpretability Rating Scale (IIRS)	The IIRS required for the imagery to be acceptable to the requestor.	O	N	3	0.0-9.9. See theater instructions…also NATO Standardization Agreement 7194 (in study) and Civil NIIRS Reference Guide.
Map Angle Constraints	True Map Angle. The true map angle is defined in the Northeast Down (NED) coordinate system with the origin at the aircraft (acft local NED) as the angle between the scene entry line of sight and the instantaneous aircraft track heading vector. The aircraft track heading vector is obtained by rotating the north unit vector of the aircraft local NED coordinate system in the aircraft local NE plane through the aircraft track heading angle. This angle is always positive.				
Min	Minimum acceptable Map Angle	O	N	5	NNN.N, 000.0-180.0 degrees
Max	Maximum acceptable Map Angle	O	N	5	NNN.N, 000.0-180.0 degrees
Grazing Angle Constraints	The angle, measured in degrees, at the target between the focus plane and the line of sight to the sensor.				
Min	Minimum acceptable Grazing Angle	O	N	4	NN.N 00.1-90.0 degrees
Max	Maximum acceptable Grazing Angle	O	N	4	NN.N, 00.1-90.0 degrees
Look Angle Constraints	The angle, measured in degrees, from True North to the line-of-sight vector from the sensor to the target tangent plane.				
Min	Minimum acceptable Look Angle	O	N	5	NNN.N, 000.0-359.9 degrees
Max	Maximum acceptable Look Angle	O	N	5	NNN.N, 000.0-359.9 degrees

Elevation Angle Constraints	The angle, measured in degrees, from the target tangent plane at the intersection of the optical line of sight with the earth's surface at the time of the first image line.				
Min	Minimum acceptable Elevation Angle	O	N/S	3	SNN, -90 to +90 degrees
Max	Maximum acceptable Elevation Angle	O	N/S	3	SNN, -90 to +90 degrees
Sun Elevation Angles	The angle, measured in degrees, from the target plane at intersection of the optical line of sight with the earth's surface at the time of the first image line.				
Min	Minimum acceptable Sun Elev Angle	O	N/S	3	SNN, -90 to +90 degrees
Max	Maximum acceptable Sun Elev Angle	O	N/S	3	SNN, -90 to +90 degrees
Sun Azimuth Angles	The angle, measured in degrees, from True North clockwise (as viewed from space) at the time of the first image line.				
Min	Minimum acceptable Sun Azimuth Angle	O	N	5	NNN.N, 000.0-359.9 degrees
Max	Maximum acceptable Sun Azimuth Angle	O	N	5	NNN.N, 000.0-359.9 degrees
Slant Range Constraints	Distance from the sensor reference point (e.g. aperture reference point) to the ground reference point.				
Min	Minimum Slant Range	O	N	10	0000000.10 - 9999999.99 (m)
Max	Maximum Slant Range	O	N	10	0000000.10 - 9999999.99 (m)
Destination Point	Numeric label for a waypoint in a navigation plan.	O	N	2	0-99
Pair Identifier	Indicates whether a collection is intended to be the initial collection of a pair, or a subsequent recollection to finish the pair.	O	A	I	I = Initial Collect. R = Recollect
Spectral Type	User defined bands - TBD	O	N	TBD	TBD
SAR					
IIRS	The IIRS required for the imagery to be acceptable to the requestor.	O	N	3	0.0-9.9. See theater instructions…also NATO Standardization Agreement 7194 (in study) and Civil NIIRS Reference Guide.
Map Angle Constraints	True Map Angle. The true map angle is defined in the NED coordinate system with the origin at the aircraft (acft local NED) as the angle between the scene entry line of sight and the instantaneous aircraft track heading vector. The aircraft track heading vector is obtained by rotating the north unit vector of the aircraft local NED coordinate system in the aircraft local NE plane through the aircraft track heading angle. This angle is always positive.				
Min	Minimum acceptable Map Angle	O	N	5	NNN.N, 000.0-180.0 degrees

					degrees
Max	Maximum acceptable Map Angle	O	N	5	NNN.N, 000.0-180.0 degrees
Grazing Angle Constraints	The angle, measured in degrees ,at the target between the focus plane and the line of sight to the sensor.				
Min	Minimum acceptable Grazing Angle	O	N	4	NN.N 00.1-90.0 degrees
Max	Maximum acceptable Grazing Angle	O	N	4	NN.N, 00.1-90.0 degrees
Look Angle Constraints	The angle, measured in degrees, from True North to the line-of-sight vector from the sensor to the target tangent plane.				
Min	Minimum acceptable Look Angle	O	N	5	NNN.N, 000.0-359.9 degrees
Max	Maximum acceptable Look Angle	O	N	5	NNN.N, 000.0-359.9 degrees
Elevation Angle Constraints	The angle, measured in degrees, from the target tangent plane at the intersection of the optical line of sight with the earth's surface at the time of the first image line.				
Min	Minimum acceptable Elevation Angle	O	N/S	3	SNN, -90 to +90 degrees
Max	Maximum acceptable Elevation Angle	O	N/S	3	SNN, -90 to +90 degrees
Slant Range Constraints	Distance from the sensor reference point (e.g. aperture reference point) to the ground reference point.				
Min	Minimum Slant Range	O	N	10	0000000.10 - 9999999.99 (m)
Max	Maximum Slant Range	O	N	10	0000000.10 - 9999999.99 (m)
Continuous Spot Angle	Required sweep angle for the continuous spot scene.	O	N	2	0-90 degrees
Destination Point	Numeric label for a waypoint in a navigation plan.	O	N	2	0-99
Pair Identifier	Indicates whether a collection is intended to be the initial collection of a pair, or a subsequent recollection to finish the pair.	O	A	1	I = Initial Collect. R = Recollect
Complex Data	Filled out if requestor wants a complex data product.	O	A	1	Y = Requestor wants complex data product. N or blank = Requestor wants detected product
Signals Intelligence (SIGINT)	TBD - SIGINT fields left empty as placeholders.				
Timing Constraints					
Collection Time Window					

Earliest	The earliest time the imagery can be acquired and still be of value to the requestor.	O	A/N/S	20	"YYYY-MM-DDThh:mm:ssZ", which is a standard xs:dateTime format with a mandatory "Z" timezone specifer on the end.
Latest	The latest time the imagery can be acquired and still be of value to the requestor.	O	A/N/S	20	"YYYY-MM-DDThh:mm:ssZ", which is a standard xs:dateTime format with a mandatory "Z" timezone specifer on the end.
Looks per Mission	The number of times the target is to be imaged during the mission.	O	N	2	00-99
Duration	The amount of time to be spent in the target area (normally for FMV or MTI), or "dwell time." Expressed in seconds.	O	N	5	1 - 86400 (24 hours)
Periodicity	The frequency of target collection a requestor needs for a given time period; expressed as the number of times a requestor wants an image in a specified time span measured in days.	O	N/S	3	N:N
MTI Revisit Interval	The time lapse, to the tenth of a second, between MTI images during collection	O	N/S	5	0000.1 - 9999.9
Collection Precedence	A unique number assigned by the collection manager to set the precedence assignment of all pre-mission tasking and during mission retasking by external requestors. It may be changed by the collection manager to respond to changing needs. The lower the assigned number value, the higher the collection precedence.	O	N	3	001-999
Exploitation Requirements					
EEI ID List	List of all user defined essential elements of information codes for the ACTM.	O	A/N	N/A	Alphanumeric: "EEI_NNNN"…size of field unbounded.
EEI Text	The essential elements of information associated with a given EEI ID. For each EEI ID, there must be text, hence this field is conditional.	C	A/N/S	N/A	Narrative. Size of field unbounded.

***Field Usage: M – Mandatory O – Optional C – Conditional**
****Type: N – Numeric A – Alpha S – Symbol**

E.3 An outline of the ACTM XML Schema

1. Aircraft Collection Tasking Message (ACTM) ACTM Metadata
    1.1. Security Classification
        1.1.1. Source
        1.1.2. <Either/Or>
            1.1.2.1. Classification String
            1.1.2.2. Classification Any
    1.2. Sensitive Compartmented Information Control Systems/Codewords
    1.3. Dissemination Control Markings
    1.4. Declassification Information
    1.5. Task Generation Timestamp
    1.6. Originator
2. Mission
    2.1. <Either/Or>
        2.1.1. Mission Number
            2.1.1.1. Program Code
            2.1.1.2. Month
            2.1.1.3. Sortie
            2.1.1.4. Year
        2.1.2. Mission Identifier
    2.2. Project Code
    2.3. Track Number
    2.4. Mission Nickname
3. Collection Requirements
    3.1. Requirement Identification (ID)
    3.2. Requirement Name
    3.3. Requestor ID
    3.4. Control Date
    3.5. Requirement Priority
    3.6. Tasks
        3.6.1. Task Priority
        3.6.2. Essential Elements of Information (EEI) ID
        3.6.3. Special Collection Instructions
        3.6.4. Target Data
            3.6.4.1. Target Type
                3.6.4.1.1. Point Target
                    3.6.4.1.1.1. Basic Encyclopedia (BE) Number
                    3.6.4.1.1.2. Target Name
                    3.6.4.1.1.3. Target Category Code
                    3.6.4.1.1.4. Center Coordinates
                        3.6.4.1.1.4.1. Latitude
                        3.6.4.1.1.4.2. Longitude
                        3.6.4.1.1.4.3. Elevation
                        3.6.4.1.1.4.4. Major Axis
                        3.6.4.1.1.4.5. Minor Axis

3.6.4.1.1.4.6. True North Angle
3.6.4.1.1.5. Look At Coordinates
3.6.4.1.1.5.1. Latitude
3.6.4.1.1.5.2. Longitude
3.6.4.1.1.5.3. Elevation
3.6.4.1.1.5.4. Major Axis
3.6.4.1.1.5.5. Minor Axis
3.6.4.1.1.5.6. True North Angle
3.6.4.1.2. Directed Search Area (DSA) Target
3.6.4.1.2.1. DSA ID
3.6.4.1.2.2. Target Name
3.6.4.1.2.3. Vertex Count
3.6.4.1.2.4. Vertex Coordinates
3.6.4.1.2.4.1. Latitude
3.6.4.1.2.4.2. Longitude
3.6.4.1.3. Line of Communication (LOC) Target
3.6.4.1.3.1. LOC ID
3.6.4.1.3.2. Target Name
3.6.4.1.3.3. Vertex Count
3.6.4.1.3.4. Default Min Swath Width
3.6.4.1.3.5. Segment Vertex
3.6.4.1.3.5.1. Latitude
3.6.4.1.3.5.2. Longitude
3.6.4.1.3.5.3. Min Segment Swath Width
3.6.4.1.3.5.4. Elevation
3.6.4.2. Area or Sector Blanking
3.6.4.2.1. Area or Sector ID
3.6.4.2.2. Area or Sector Name
3.6.4.2.3. Vertex Count
3.6.4.2.4. Vertex Coordinates
3.6.4.2.4.1. Latitude
3.6.4.2.4.2. Longitude
3.6.4.3. <Either/Or>
3.6.4.3.1. Country Code2
3.6.4.3.2. Country Code3
3.6.4.4. Geographic Region
3.6.5. Collection Constraints
3.6.5.1. Sensor ID
3.6.5.2. Sensor Mode
3.6.5.3. Sensor Specific Data
3.6.5.3.1. Optical
3.6.5.3.1.1. Imagery Interpretability Rating Scale (IIRS)
3.6.5.3.1.2. Map Angle Constraints
3.6.5.3.1.2.1. Min
3.6.5.3.1.2.2. Max
3.6.5.3.1.3. Grazing Angle Constraints
3.6.5.3.1.3.1. Min

3.6.5.3.1.3.2. Max
3.6.5.3.1.4. Look Angle Constraints
3.6.5.3.1.4.1. Min
3.6.5.3.1.4.2. Max
3.6.5.3.1.5. Elevation Angle Constraints
3.6.5.3.1.5.1. Min
3.6.5.3.1.5.2. Max
3.6.5.3.1.6. Sun Elevation Angles
3.6.5.3.1.6.1. Min
3.6.5.3.1.6.2. Max
3.6.5.3.1.7. Sun Azimuth Angles
3.6.5.3.1.7.1. Min
3.6.5.3.1.7.2. Max
3.6.5.3.1.8. Slant Range Constraints
3.6.5.3.1.8.1. Min
3.6.5.3.1.8.2. Max
3.6.5.3.1.9. Destination Point
3.6.5.3.1.10. Pair Identifier
3.6.5.3.1.11. Spectral Type
3.6.5.3.2. Synthetic Aperture Radar
3.6.5.3.2.1. IIRS
3.6.5.3.2.2. Map Angle Constraints
3.6.5.3.2.2.1. Min
3.6.5.3.2.2.2. Max
3.6.5.3.2.3. Grazing Angle Constraints
3.6.5.3.2.3.1. Min
3.6.5.3.2.3.2. Max
3.6.5.3.2.4. Look Angle Constraints
3.6.5.3.2.4.1. Min
3.6.5.3.2.4.2. Max
3.6.5.3.2.5. Elevation Angle Constraints
3.6.5.3.2.5.1. Min
3.6.5.3.2.5.2. Max
3.6.5.3.2.6. Slant Range Constraints
3.6.5.3.2.6.1. Min
3.6.5.3.2.6.2. Max
3.6.5.3.2.7. Continuous Spot Angle
3.6.5.3.2.8. Destination Point
3.6.5.3.2.9. Pair Identifier
3.6.5.3.2.10. Complex Data
3.6.5.3.3. Signals Intelligence
3.6.6. Timing Constraints
3.6.6.1. Collection Time Window
3.6.6.1.1. Earliest
3.6.6.1.2. Latest
3.6.6.2. Looks per Mission
3.6.6.3. Duration

3.6.6.4.MTI Revisit Interval
3.6.6.5.Periodicity
3.7. Collection Precedence
4. Exploitation Requirements
4.1.1. EEI ID List
4.1.2. EEI Text

E.4 The ACTM XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v2004 rel. 3 U (http://www.xmlspy.com) by
Austin Burge (RAYTHEON SYSTEMS COMPANY) -->
<!-- edited with XMLSpy v2006 U (http://www.altova.com) by Diana Seiler
(SeiCorp) -->
<!-- edited by hand by Ken Bateman -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:simpleType name="String35">
    <xs:restriction base="xs:string">
      <xs:maxLength value="35"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="String15">
    <xs:restriction base="xs:string">
      <xs:maxLength value="15"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="EEI_ID">
    <xs:restriction base="xs:ID">
      <xs:pattern value="EEI_[0-9]+"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="EEI_IDREF">
    <xs:restriction base="xs:IDREF">
      <xs:pattern value="EEI_[0-9]+"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="IIRSType">
    <xs:annotation>
      <xs:documentation>Must be digit, dot, digit, like "3.2", range
0.0 to 9.9</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:decimal">
      <xs:pattern value="[0-9]\.[0-9]"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="AltitudeBaseType">
    <xs:restriction base="xs:integer"/>
  </xs:simpleType>
  <xs:complexType name="AltitudeType">
    <xs:simpleContent>
      <xs:extension base="AltitudeBaseType">
        <xs:attribute name="units" use="required" fixed="meters"/>
      </xs:extension>
```



```
    </xs:simpleContent>
  </xs:complexType>
  <xs:simpleType name="decimal4dot1">
    <xs:annotation>
      <xs:documentation>Four digits, then a decimal point, then a
tenths digit.  Range 0</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:decimal">
      <xs:minInclusive value="0000.1"/>
      <xs:pattern value="[0-9][0-9][0-9][0-9]\.[0-9]"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="LatitudeDMSH">
    <xs:annotation>
      <xs:documentation>"DDMMSS", then optional decimal point and
decimal digits ".SSS", then N or S.</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:pattern value="[0-9][0-9][0-5][0-9][0-5][0-9](\.[0-
9]+)?[NS]"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="LongitudeDMSH">
    <xs:annotation>
      <xs:documentation>"DDDMMSS", then optional decimal point and
decimal digits ".SSS", then E or W.</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:pattern value="[01][0-9][0-9][0-5][0-9][0-5][0-9](\.[0-
9]+)?[EW]"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="UTCDate">
    <xs:annotation>
      <xs:documentation>Standard XML date, except required to
explicitly have "Z" zulu specifier at end.</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:date">
      <xs:pattern value=".*Z"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="UTCDateTime">
    <xs:annotation>
      <xs:documentation>Standard XML dateTime, except required to
explicitly have "Z" zulu specifier at end.</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:dateTime">
      <xs:pattern value=".*Z"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="MissionNumberType">
    <xs:sequence>
      <xs:element name="ProgramCode">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:pattern value="([0-8A-Z][A-Z]|9[0-9A-Y])">
              <xs:annotation>
```



```
                <xs:documentation>2 alphanumeric, do not use
9Z</xs:documentation>
              </xs:annotation>
            </xs:pattern>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="Month_int">
        <xs:simpleType>
          <xs:restriction base="xs:integer">
            <xs:pattern value="(0?[1-9]|1[0-2])">
              <xs:annotation>
                <xs:documentation>01-12 or 1-12</xs:documentation>
              </xs:annotation>
            </xs:pattern>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="Sortie_int">
        <xs:simpleType>
          <xs:restriction base="xs:integer">
            <xs:minInclusive value="01"/>
            <xs:pattern value="[0-9]{2}">
              <xs:annotation>
                <xs:documentation>Must be 2 digits, 01-
99</xs:documentation>
              </xs:annotation>
            </xs:pattern>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="Year_int">
        <xs:simpleType>
          <xs:restriction base="xs:integer">
            <xs:pattern value="[0-9]{4}">
              <xs:annotation>
                <xs:documentation>4 digits, no explicit constraint on
range</xs:documentation>
              </xs:annotation>
            </xs:pattern>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
  <xs:simpleType name="MissionIdentifierType">
    <xs:restriction base="xs:string">
      <xs:maxLength value="54"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="UTCTime">
    <xs:annotation>
      <xs:documentation>Standard XML time, except required to
explicitly have "Z" zulu specifier at end.</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:time">
      <xs:pattern value=".*Z"/>
    </xs:restriction>
```



```
  </xs:simpleType>
  <xs:simpleType name="DurationBaseType">
    <xs:annotation>
      <xs:documentation>0 to 86,400 seconds per day</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:integer">
      <xs:minInclusive value="0"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="DurationType">
    <xs:simpleContent>
      <xs:extension base="DurationBaseType">
        <xs:attribute name="units" use="required" fixed="seconds"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:complexType name="GroundPointLocation">
    <xs:sequence>
      <xs:element name="Latitude_DMSH" type="LatitudeDMSH"/>
      <xs:element name="Longitude_DMSH" type="LongitudeDMSH"/>
      <xs:element name="Elevation_m" type="xs:decimal" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
  <xs:simpleType name="SlantRangeLength">
    <xs:annotation>
      <xs:documentation>Slant range, 0.1 to 9999999.99, decimals
optional</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:decimal">
      <xs:minInclusive value="0.1"/>
      <xs:maxInclusive value="9999999.99"/>
      <xs:pattern value="[0-9]+(\.[0-9]+)?"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="SwathWidth">
    <xs:annotation>
      <xs:documentation>Width of LOC swath, from 1 to 999999
meters</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:integer">
      <xs:minInclusive value="1"/>
      <xs:maxInclusive value="999999"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="SwathGroundPointLocation">
    <xs:sequence>
      <xs:element name="Latitude_DMSH" type="LatitudeDMSH"/>
      <xs:element name="Longitude_DMSH" type="LongitudeDMSH"/>
      <xs:element name="MinSwathWidth" type="SwathWidth" minOccurs="0">
        <xs:annotation>
          <xs:documentation>This element overrides DefaultMinSwathWidth
for only the segment following this vertex</xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="Elevation_m" type="xs:decimal" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="Number" type="xs:integer" use="required"/>
```

```
  </xs:complexType>
  <xs:complexType name="DSAGroundPointLocation">
    <xs:sequence>
      <xs:element name="Latitude_DMSH" type="LatitudeDMSH"/>
      <xs:element name="Longitude_DMSH" type="LongitudeDMSH"/>
    </xs:sequence>
  </xs:complexType>
  <xs:simpleType name="AxisLength">
    <xs:annotation>
      <xs:documentation>Length of axis for GroundEllipseLocation, 0.1
to 9999.9, decimals optional</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:decimal">
      <xs:minInclusive value="0.1"/>
      <xs:maxInclusive value="9999.9"/>
      <xs:pattern value="[0-9]+(\.[0-9]+)?"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="GroundEllipseLocation">
    <xs:sequence>
      <xs:element name="Latitude_DMSH" type="LatitudeDMSH"/>
      <xs:element name="Longitude_DMSH" type="LongitudeDMSH"/>
      <xs:element name="Elevation_m" minOccurs="0">
        <xs:simpleType>
          <xs:restriction base="xs:integer">
            <xs:minInclusive value="-00680"/>
            <xs:maxInclusive value="+10688"/>
            <xs:pattern value="[-+][0-9]{5}"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="MajorAxis_m" type="AxisLength" minOccurs="0"/>
      <xs:element name="MinorAxis_m" type="AxisLength" minOccurs="0"/>
      <xs:element name="TrueNorthAngle_deg" minOccurs="0">
        <xs:simpleType>
          <xs:restriction base="xs:decimal">
            <xs:minInclusive value="0"/>
            <xs:maxInclusive value="180"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="AngleRange-90to90">
    <xs:sequence>
      <xs:element name="Min_deg">
        <xs:simpleType>
          <xs:restriction base="xs:decimal">
            <xs:minInclusive value="-90"/>
            <xs:maxInclusive value="90"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="Max_deg">
        <xs:simpleType>
          <xs:restriction base="xs:decimal">
            <xs:minInclusive value="-90"/>
            <xs:maxInclusive value="90"/>
```



```
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="AngleRange90">
  <xs:sequence>
    <xs:element name="Min_deg">
      <xs:simpleType>
        <xs:restriction base="xs:decimal">
          <xs:minInclusive value="0"/>
          <xs:maxInclusive value="90"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
    <xs:element name="Max_deg">
      <xs:simpleType>
        <xs:restriction base="xs:decimal">
          <xs:minInclusive value="0"/>
          <xs:maxInclusive value="90"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="AngleRange180">
  <xs:sequence>
    <xs:element name="Min_deg">
      <xs:simpleType>
        <xs:restriction base="xs:decimal">
          <xs:minInclusive value="0"/>
          <xs:maxInclusive value="180"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
    <xs:element name="Max_deg">
      <xs:simpleType>
        <xs:restriction base="xs:decimal">
          <xs:minInclusive value="0"/>
          <xs:maxInclusive value="180"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="AngleRange360">
  <xs:sequence>
    <xs:element name="Min_deg">
      <xs:simpleType>
        <xs:restriction base="xs:decimal">
          <xs:minInclusive value="0"/>
          <xs:maxInclusive value="360"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
    <xs:element name="Max_deg">
      <xs:simpleType>
```

```
          <xs:restriction base="xs:decimal">
            <xs:minInclusive value="0"/>
            <xs:maxInclusive value="360"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="AircraftCollectionTaskMessageV2.2">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="ACTMMetadata">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="SecurityClassification">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="Source">
                      <xs:simpleType>
                        <xs:restriction base="xs:string">
                          <xs:length value="3"/>
                        </xs:restriction>
                      </xs:simpleType>
                    </xs:element>
                    <xs:choice>
                      <xs:element name="Classification_string">
                        <xs:simpleType>
                          <xs:restriction base="xs:string">
                            <xs:maxLength value="128"/>
                          </xs:restriction>
                        </xs:simpleType>
                      </xs:element>
                      <xs:element name="Classification_any">
                        <xs:complexType>
                          <xs:sequence>
                            <xs:any>
                              <xs:annotation>
                                <xs:documentation>
                                  The Classification tag uses the
xs:any schema element, which means that this tag may contain any string
or as-yet-undefined XML element, to accomodate current and future
classification markings.
                                </xs:documentation>
                              </xs:annotation>
                            </xs:any>
                          </xs:sequence>
                        </xs:complexType>
                      </xs:element>
                    </xs:choice>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element name="SCIControlSystemsCodewords"
type="String35" minOccurs="0"/>
              <xs:element name="DisseminationControlMarkings"
type="String35" minOccurs="0"/>
              <xs:element name="DeclassificationInformation"
type="String35" minOccurs="0"/>
```



```
              <xs:element name="TaskGenerationTimestamp"
type="UTCDateTime" minOccurs="0"/>
              <xs:element name="Originator" type="xs:string"
minOccurs="0"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="Mission">
          <xs:complexType>
            <xs:sequence>
              <xs:choice>
                <xs:element name="MissionNumber"
type="MissionNumberType"/>
                <xs:element name="MissionIdentifier"
type="MissionIdentifierType"/>
              </xs:choice>
              <xs:element name="ProjectCode" minOccurs="0">
                <xs:simpleType>
                  <xs:restriction base="xs:string">
                    <xs:pattern value="([A-Y][A-Z]|Z[A-Y])">
                      <xs:annotation>
                        <xs:documentation>2 letter code, not
ZZ</xs:documentation>
                      </xs:annotation>
                    </xs:pattern>
                  </xs:restriction>
                </xs:simpleType>
              </xs:element>
              <xs:element name="TrackNumber" minOccurs="0">
                <xs:simpleType>
                  <xs:restriction base="xs:string">
                    <xs:pattern value="[A-Z0-9]{6}">
                      <xs:annotation>
                        <xs:documentation>6-char
alphanumeric</xs:documentation>
                      </xs:annotation>
                    </xs:pattern>
                  </xs:restriction>
                </xs:simpleType>
              </xs:element>
              <xs:element name="MissionNickname" type="xs:string"
minOccurs="0"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="CollectionRequirements"
maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="RequirementID">
                <xs:simpleType>
                  <xs:restriction base="xs:string">
                    <xs:maxLength value="12"/>
                  </xs:restriction>
                </xs:simpleType>
              </xs:element>
              <xs:element name="RequirementName" minOccurs="0">
```



```
                <xs:simpleType>
                  <xs:restriction base="xs:string">
                    <xs:maxLength value="50"/>
                  </xs:restriction>
                </xs:simpleType>
              </xs:element>
              <xs:element name="RequestorID" type="xs:string"
minOccurs="0"/>
              <xs:element name="ControlDate_UTCDateTime"
type="UTCDateTime"/>
              <xs:element name="RequirementPriority_int">
                <xs:simpleType>
                  <xs:restriction base="xs:integer">
                    <xs:minInclusive value="0"/>
                    <xs:maxInclusive value="999"/>
                  </xs:restriction>
                </xs:simpleType>
              </xs:element>
              <xs:element name="Tasks" maxOccurs="unbounded">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="TaskPriority_int">
                      <xs:simpleType>
                        <xs:restriction base="xs:integer">
                          <xs:minInclusive value="0"/>
                          <xs:maxInclusive value="999"/>
                        </xs:restriction>
                      </xs:simpleType>
                    </xs:element>
                    <xs:element name="EEI_ID" type="EEI_IDREF"
minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element name="SpecialCollectionInstructions"
type="xs:string" minOccurs="0"/>
                    <xs:element name="TargetData">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="TargetType">
                            <xs:complexType>
                              <xs:choice>
                                <xs:element name="PointTarget">
                                  <xs:complexType>
                                    <xs:sequence>
                                      <xs:element name="BENumber"
minOccurs="0">
                                        <xs:simpleType>
                                          <xs:restriction
base="xs:string">
                                            <xs:annotation>
                                              <xs:documentation>Either
10 or 15 letters, numbers, and symbols</xs:documentation>
                                            </xs:annotation>
                                            <xs:pattern
value=".{10}(.{5})?"/>
                                          </xs:restriction>
                                        </xs:simpleType>
                                      </xs:element>
                                      <xs:element name="TargetName"
type="xs:string"/>
```



```
                                      <xs:element
name="Target_CategoryCode_int" minOccurs="0">
                                        <xs:simpleType>
                                          <xs:restriction
base="xs:integer">

                                            <xs:minInclusive
value="0"/>
                                          </xs:restriction>
                                        </xs:simpleType>
                                      </xs:element>
                                      <xs:element name="CenterCoords"
type="GroundEllipseLocation"/>
                                      <xs:element name="LookAtCoords"
type="GroundEllipseLocation" minOccurs="0"/>
                                    </xs:sequence>
                                  </xs:complexType>
                                </xs:element>
                                <xs:element name="DSATarget">
                                  <xs:complexType>
                                    <xs:sequence>
                                      <xs:element name="DSA_ID"
type="String15" minOccurs="0"/>
                                      <xs:element name="TargetName">
                                        <xs:simpleType>
                                          <xs:restriction
base="xs:string">

                                            <xs:maxLength value="38"/>
                                          </xs:restriction>
                                        </xs:simpleType>
                                      </xs:element>
                                      <xs:element
name="VertexCount_int">

                                        <xs:simpleType>
                                          <xs:restriction
base="xs:integer">

                                            <xs:minInclusive
value="3"/>

                                            <xs:maxInclusive
value="24"/>
                                          </xs:restriction>
                                        </xs:simpleType>
                                      </xs:element>
                                      <xs:element name="VertexCoords"
minOccurs="3" maxOccurs="24">
                                        <xs:complexType>
                                          <xs:complexContent>
                                            <xs:extension
base="DSAGroundPointLocation">
                                              <xs:attribute
name="Number" type="xs:integer" use="required"/>
                                            </xs:extension>
                                          </xs:complexContent>
                                        </xs:complexType>
                                      </xs:element>
                                    </xs:sequence>
                                  </xs:complexType>
                                </xs:element>
```

```
                                <xs:element name="LOCTarget">
                                  <xs:complexType>
                                    <xs:sequence>
                                      <xs:element name="LOC_ID"
type="String15" minOccurs="0"/>
                                      <xs:element name="TargetName">
                                        <xs:simpleType>
                                          <xs:restriction
base="xs:string">

                                            <xs:maxLength value="38"/>
                                          </xs:restriction>
                                        </xs:simpleType>
                                      </xs:element>
                                      <xs:element
name="VertexCount_int">

                                        <xs:simpleType>
                                          <xs:restriction
base="xs:integer">

                                            <xs:minInclusive
value="2"/>

                                            <xs:maxInclusive
value="24"/>

                                          </xs:restriction>
                                        </xs:simpleType>
                                      </xs:element>
                                      <xs:element
name="DefaultMinSwathWidth" type="SwathWidth" minOccurs="0"/>
                                      <xs:element name="SegmentVertex"
type="SwathGroundPointLocation" minOccurs="2" maxOccurs="24"/>
                                    </xs:sequence>
                                  </xs:complexType>
                                </xs:element>
                              </xs:choice>
                            </xs:complexType>
                          </xs:element>
                          <xs:element name="AreaOrSectorBlanking"
minOccurs="0">
                            <xs:complexType>
                              <xs:sequence>
                                <xs:element name="AreaOrSectorID"
type="xs:string" minOccurs="0"/>
                                <xs:element name="AreaOrSectorName"
type="xs:string"/>
                                <xs:element name="VertexCoords"
minOccurs="3" maxOccurs="24">
                                  <xs:complexType>
                                    <xs:complexContent>
                                      <xs:extension
base="DSAGroundPointLocation">

                                        <xs:attribute name="Number"
type="xs:integer" use="required"/>
                                      </xs:extension>
                                    </xs:complexContent>
                                  </xs:complexType>
                                </xs:element>
                                <xs:element name="VertexCount_int">
                                  <xs:simpleType>
                                    <xs:restriction base="xs:integer">
```

```
                                      <xs:minInclusive value="3"/>
                                      <xs:maxInclusive value="24"/>
                                    </xs:restriction>
                                  </xs:simpleType>
                                </xs:element>
                              </xs:sequence>
                            </xs:complexType>
                          </xs:element>
                          <xs:choice minOccurs="0">
                            <xs:element name="CountryCode2"
minOccurs="0" maxOccurs="unbounded">
                              <xs:simpleType>
                                <xs:restriction base="xs:string">
                                  <xs:pattern value="[A-Z]{2}"/>
                                </xs:restriction>
                              </xs:simpleType>
                            </xs:element>
                            <xs:element name="CountryCode3"
minOccurs="0" maxOccurs="unbounded">
                              <xs:simpleType>
                                <xs:restriction base="xs:string">
                                  <xs:pattern value="[A-Z]{3}"/>
                                </xs:restriction>
                              </xs:simpleType>
                            </xs:element>
                          </xs:choice>
                          <xs:element name="GeographicRegion"
type="xs:string" minOccurs="0"/>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                    <xs:element name="Collection_Constraints">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="SensorID">
                            <xs:simpleType>
                              <xs:restriction base="xs:string">
                                <xs:maxLength value="50"/>
                              </xs:restriction>
                            </xs:simpleType>
                          </xs:element>
                          <xs:element name="SensorMode" minOccurs="0">
                            <xs:simpleType>
                              <xs:restriction base="xs:string">
                                <xs:maxLength value="50"/>
                              </xs:restriction>
                            </xs:simpleType>
                          </xs:element>
                          <xs:element name="SensorSpecificData"
minOccurs="0">
                            <xs:complexType>
                              <xs:choice>
                                <xs:element name="Optical"
minOccurs="0">
                                  <xs:complexType>
                                    <xs:sequence>
```



```
                                      <xs:element name="IIRS"
type="IIRSType" minOccurs="0"/>
                                      <xs:element
name="MapAngleConstraints" minOccurs="0">
                                        <xs:complexType>
                                          <xs:complexContent>
                                            <xs:extension
base="AngleRange180">
                                              <xs:attribute
name="Absolute" use="optional">
                                                <xs:simpleType>
                                                  <xs:restriction
base="xs:string">
                                                    <xs:enumeration
value="True"/>
                                                    <xs:enumeration
value="False"/>
                                                  </xs:restriction>
                                                </xs:simpleType>
                                              </xs:attribute>
                                            </xs:extension>
                                          </xs:complexContent>
                                        </xs:complexType>
                                      </xs:element>
                                      <xs:element
name="GrazingAngleConstraints" minOccurs="0">
                                        <xs:complexType>
                                          <xs:complexContent>
                                            <xs:extension
base="AngleRange90">
                                              <xs:attribute
name="Absolute" use="optional">
                                                <xs:simpleType>
                                                  <xs:restriction
base="xs:string">
                                                    <xs:enumeration
value="True"/>
                                                    <xs:enumeration
value="False"/>
                                                  </xs:restriction>
                                                </xs:simpleType>
                                              </xs:attribute>
                                            </xs:extension>
                                          </xs:complexContent>
                                        </xs:complexType>
                                      </xs:element>
                                      <xs:element
name="LookAngleConstraints" minOccurs="0">
                                        <xs:complexType>
                                          <xs:complexContent>
                                            <xs:extension
base="AngleRange360">
                                              <xs:attribute
name="Absolute" use="optional">
                                                <xs:simpleType>
                                                  <xs:restriction
base="xs:string">
```

```
                                                    <xs:enumeration
value="True"/>
                                                    <xs:enumeration
value="False"/>
                                                  </xs:restriction>
                                                </xs:simpleType>
                                              </xs:attribute>
                                            </xs:extension>
                                          </xs:complexContent>
                                        </xs:complexType>
                                      </xs:element>
                                      <xs:element
name="ElevationAngleConstraints" minOccurs="0">
                                        <xs:complexType>
                                          <xs:complexContent>
                                            <xs:extension
base="AngleRange-90to90">
                                              <xs:attribute
name="Absolute" use="optional">
                                                <xs:simpleType>
                                                  <xs:restriction
base="xs:string">
                                                    <xs:enumeration
value="True"/>
                                                    <xs:enumeration
value="False"/>
                                                  </xs:restriction>
                                                </xs:simpleType>
                                              </xs:attribute>
                                            </xs:extension>
                                          </xs:complexContent>
                                        </xs:complexType>
                                      </xs:element>
                                      <xs:element name="SunElevAngles"
minOccurs="0">
                                        <xs:complexType>
                                          <xs:complexContent>
                                            <xs:extension
base="AngleRange-90to90">
                                              <xs:attribute
name="Absolute" use="optional">
                                                <xs:simpleType>
                                                  <xs:restriction
base="xs:string">
                                                    <xs:enumeration
value="True"/>
                                                    <xs:enumeration
value="False"/>
                                                  </xs:restriction>
                                                </xs:simpleType>
                                              </xs:attribute>
                                            </xs:extension>
                                          </xs:complexContent>
                                        </xs:complexType>
                                      </xs:element>
                                      <xs:element
name="SunAzimuthAngles" minOccurs="0">
```



```
                                        <xs:complexType>
                                          <xs:complexContent>
                                            <xs:extension
base="AngleRange360">

                                              <xs:attribute
name="Absolute" use="optional">
                                                <xs:simpleType>
                                                  <xs:restriction
base="xs:string">

                                                    <xs:enumeration
value="True"/>

                                                    <xs:enumeration
value="False"/>

                                                  </xs:restriction>
                                                </xs:simpleType>
                                              </xs:attribute>
                                            </xs:extension>
                                          </xs:complexContent>
                                        </xs:complexType>
                                      </xs:element>
                                      <xs:element
name="SlantRangeConstraints" minOccurs="0">
                                        <xs:complexType>
                                          <xs:sequence>
                                            <xs:element name="Min"
type="SlantRangeLength"/>

                                            <xs:element name="Max"
type="SlantRangeLength"/>
                                          </xs:sequence>
                                          <xs:attribute name="Absolute"
use="optional">

                                            <xs:simpleType>
                                              <xs:restriction
base="xs:string">

                                                <xs:enumeration
value="True"/>

                                                <xs:enumeration
value="False"/>

                                              </xs:restriction>
                                            </xs:simpleType>
                                          </xs:attribute>
                                        </xs:complexType>
                                      </xs:element>
                                      <xs:element
name="DestinationPoint" minOccurs="0">
                                        <xs:simpleType>
                                          <xs:restriction
base="xs:integer">

                                            <xs:minInclusive
value="0"/>

                                          </xs:restriction>
                                        </xs:simpleType>
                                      </xs:element>
                                      <xs:element name="PairIdentifier"
minOccurs="0">

                                        <xs:simpleType>
                                          <xs:annotation>
```

```
                                            <xs:documentation>Single
letter, "I" = Initial Collect, "R" = Recollect</xs:documentation>
                                          </xs:annotation>
                                          <xs:restriction
base="xs:string">
                                            <xs:enumeration value="I"/>
                                            <xs:enumeration value="R"/>
                                          </xs:restriction>
                                        </xs:simpleType>
                                      </xs:element>
                                      <xs:element name="SpectralType"
type="xs:string" minOccurs="0">
                                        <xs:annotation>
                                          <xs:documentation>To Be
Discussed</xs:documentation>
                                        </xs:annotation>
                                      </xs:element>
                                    </xs:sequence>
                                  </xs:complexType>
                                </xs:element>
                                <xs:element name="SAR" minOccurs="0">
                                  <xs:complexType>
                                    <xs:sequence>
                                      <xs:element name="IIRS"
type="IIRSType" minOccurs="0"/>
                                      <xs:element
name="MapAngleConstraints" minOccurs="0">
                                        <xs:complexType>
                                          <xs:complexContent>
                                            <xs:extension
base="AngleRange180">
                                              <xs:attribute
name="Absolute" use="optional">
                                                <xs:simpleType>
                                                  <xs:restriction
base="xs:string">
                                                    <xs:enumeration
value="True"/>
                                                    <xs:enumeration
value="False"/>
                                                  </xs:restriction>
                                                </xs:simpleType>
                                              </xs:attribute>
                                            </xs:extension>
                                          </xs:complexContent>
                                        </xs:complexType>
                                      </xs:element>
                                      <xs:element
name="GrazingAngleConstraints" minOccurs="0">
                                        <xs:complexType>
                                          <xs:complexContent>
                                            <xs:extension
base="AngleRange90">
                                              <xs:attribute
name="Absolute" use="optional">
                                                <xs:simpleType>
```



```
                                                  <xs:restriction
base="xs:string">
                                                    <xs:enumeration
value="True"/>
                                                    <xs:enumeration
value="False"/>
                                                  </xs:restriction>
                                                </xs:simpleType>
                                              </xs:attribute>
                                            </xs:extension>
                                          </xs:complexContent>
                                        </xs:complexType>
                                      </xs:element>
                                      <xs:element
name="LookAngleConstraints" minOccurs="0">
                                        <xs:complexType>
                                          <xs:complexContent>
                                            <xs:extension
base="AngleRange360">
                                              <xs:attribute
name="Absolute" use="optional">
                                                <xs:simpleType>
                                                  <xs:restriction
base="xs:string">
                                                    <xs:enumeration
value="True"/>
                                                    <xs:enumeration
value="False"/>
                                                  </xs:restriction>
                                                </xs:simpleType>
                                              </xs:attribute>
                                            </xs:extension>
                                          </xs:complexContent>
                                        </xs:complexType>
                                      </xs:element>
                                      <xs:element
name="ElevationAngleConstraints" minOccurs="0">
                                        <xs:complexType>
                                          <xs:complexContent>
                                            <xs:extension
base="AngleRange-90to90">
                                              <xs:attribute
name="Absolute" use="optional">
                                                <xs:simpleType>
                                                  <xs:restriction
base="xs:string">
                                                    <xs:enumeration
value="True"/>
                                                    <xs:enumeration
value="False"/>
                                                  </xs:restriction>
                                                </xs:simpleType>
                                              </xs:attribute>
                                            </xs:extension>
                                          </xs:complexContent>
                                        </xs:complexType>
                                      </xs:element>
```



```
                                      <xs:element
name="SlantRangeConstraints" minOccurs="0">
                                        <xs:complexType>
                                          <xs:sequence>
                                            <xs:element name="Min"
type="SlantRangeLength"/>
                                            <xs:element name="Max"
type="SlantRangeLength"/>
                                          </xs:sequence>
                                          <xs:attribute name="Absolute"
use="optional">
                                            <xs:simpleType>
                                              <xs:restriction
base="xs:string">
                                                <xs:enumeration
value="True"/>
                                                <xs:enumeration
value="False"/>
                                              </xs:restriction>
                                            </xs:simpleType>
                                          </xs:attribute>
                                        </xs:complexType>
                                      </xs:element>
                                      <xs:element
name="ContinuousSpotAngle_int" minOccurs="0">
                                        <xs:simpleType>
                                          <xs:restriction
base="xs:integer">
                                            <xs:minInclusive
value="0"/>
                                            <xs:maxInclusive
value="90"/>
                                          </xs:restriction>
                                        </xs:simpleType>
                                      </xs:element>
                                      <xs:element
name="DestinationPointP" minOccurs="0">
                                        <xs:simpleType>
                                          <xs:restriction
base="xs:integer">
                                            <xs:minInclusive
value="0"/>
                                          </xs:restriction>
                                        </xs:simpleType>
                                      </xs:element>
                                      <xs:element name="PairIdentifier"
minOccurs="0">
                                        <xs:simpleType>
                                          <xs:annotation>
                                            <xs:documentation>Single
letter</xs:documentation>
                                          </xs:annotation>
                                          <xs:restriction
base="xs:string">
                                            <xs:enumeration value="I"/>
                                            <xs:enumeration value="R"/>
                                          </xs:restriction>
```



```
                                        </xs:simpleType>
                                      </xs:element>
                                      <xs:element name="ComplexData"
minOccurs="0">
                                        <xs:simpleType>
                                          <xs:restriction
base="xs:string">
                                            <xs:enumeration value="Y"/>
                                            <xs:enumeration value="N"/>
                                          </xs:restriction>
                                        </xs:simpleType>
                                      </xs:element>
                                    </xs:sequence>
                                  </xs:complexType>
                                </xs:element>
                                <xs:element name="SIGINT"
type="xs:string" minOccurs="0">
                                  <xs:annotation>
                                    <xs:documentation>Placeholder for
future use</xs:documentation>
                                  </xs:annotation>
                                </xs:element>
                              </xs:choice>
                            </xs:complexType>
                          </xs:element>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                    <xs:element name="TimingConstraints" minOccurs="0">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="CollectionTimeWindow"
minOccurs="0">
                            <xs:complexType>
                              <xs:sequence>
                                <xs:element name="Earliest_UTCDateTime"
type="UTCDateTime" minOccurs="0"/>
                                <xs:element name="Latest_UTCDateTime"
type="UTCDateTime" minOccurs="0"/>
                              </xs:sequence>
                              <xs:attribute name="Absolute"
use="optional">
                                <xs:simpleType>
                                  <xs:restriction base="xs:string">
                                    <xs:enumeration value="True"/>
                                    <xs:enumeration value="False"/>
                                  </xs:restriction>
                                </xs:simpleType>
                              </xs:attribute>
                            </xs:complexType>
                          </xs:element>
                          <xs:element name="LooksPerMission_int"
minOccurs="0">
                            <xs:simpleType>
                              <xs:restriction base="xs:integer">
                                <xs:minInclusive value="0"/>
                                <xs:maxInclusive value="99"/>
                              </xs:restriction>
```



```
                          </xs:simpleType>
                        </xs:element>
                        <xs:element name="Duration"
type="DurationType" minOccurs="0"/>
                        <xs:element name="MTIRevisitInterval"
type="decimal4dot1" minOccurs="0"/>
                        <xs:element name="Periodicity" type="xs:time"
minOccurs="0"/>
                      </xs:sequence>
                    </xs:complexType>
                  </xs:element>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
            <xs:element name="CollectionPrecedence_int"
minOccurs="0">
              <xs:simpleType>
                <xs:restriction base="xs:integer">
                  <xs:minInclusive value="1"/>
                  <xs:maxInclusive value="999"/>
                </xs:restriction>
              </xs:simpleType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="ExploitationRequirements" minOccurs="0"
maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="EEI_ID" type="EEI_ID"/>
            <xs:element name="EEI_Text" type="xs:string"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```


Bibliography

FM34-2. (1994). CollectionManagement and Synchronization Planning.

Fujimoto, R. M. (1999). Parallel and distributed simulation systems Reihe Wiley Series on Parallel and Distributed Computing. New York, Chichester, Weinheim, Brisbane, Singapore, Toronto: John Wiley & Sons, Inc.

Klein, M. H. A Practitioner’s Handbook for Real-Time Analysis: Kluwer Academic Press.

Kleindorfer, G. B., O'Neill, L. und Ganeshan, R. (1998). Validation in Simulation: Various Positions in the Philosophy of Science. Management Science, 44(8), 1087-1099.

Oreskes, N., Shrader-Frechette, K. und Belitz, K. (1994). Verification, validation, and confirmation of numerical models in the earth sciences. Science, 263, 641-646.

Zeigler, B. P., Praehofer, H. und Kim, T. G. (2000). Theory of modeling and simulation: integrating discrete event and continuous complex dynamic systems, Bd. 2. San Diego: Academic Press.

