Open Geospatial Consortium

Submission Date: 2014-02-24

Approval Date: 2014-07-10

Publication Date: 2014-07-17

External identifier of this OGC® document: http://www.opengis.net/def/BP/rest-sps-for-eo-tasking/1.0

Internal reference number of this OGC® document: OGC 14-012r1

Version: 1.0

Category: OGC® Best Practice

Editor: Nicolas FANJEAU
Sebastian ULRICH

OGC RESTful encoding of OGC Sensor Planning Service for Earth Observation satellite Tasking

Warning

This document defines an OGC Best Practices on a particular technology or approach related to an OGC standard. This document is not an OGC Standard and may not be referred to as an OGC Standard. It is subject to change without notice. However, this document is an official position of the OGC membership on this particular technology topic.

Document type: OGC® Best Practice
Document subtype: if applicable
Document stage: Approved for public release
Document language: English

License Agreement

Permission is hereby granted by the Open Geospatial Consortium, ("Licensor"), free of charge and subject to the terms set forth below, to any person obtaining a copy of this Intellectual Property and any associated documentation, to deal in the Intellectual Property without restriction (except as set forth below), including without limitation the rights to implement, use, copy, modify, merge, publish, distribute, and/or sublicense copies of the Intellectual Property, and to permit persons to whom the Intellectual Property is furnished to do so, provided that all copyright notices on the intellectual property are retained intact and that each person to whom the Intellectual Property is furnished agrees to the terms of this Agreement.

If you modify the Intellectual Property, all copies of the modified Intellectual Property must include, in addition to the above copyright notice, a notice that the Intellectual Property includes modifications that have not been approved or adopted by LICENSOR.

THIS LICENSE IS A COPYRIGHT LICENSE ONLY, AND DOES NOT CONVEY ANY RIGHTS UNDER ANY PATENTS THAT MAY BE IN FORCE ANYWHERE IN THE WORLD.

THE INTELLECTUAL PROPERTY IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE DO NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE INTELLECTUAL PROPERTY WILL MEET YOUR REQUIREMENTS OR THAT THE OPERATION OF THE INTELLECTUAL PROPERTY WILL BE UNINTERRUPTED OR ERROR FREE. ANY USE OF THE INTELLECTUAL PROPERTY SHALL BE MADE ENTIRELY AT THE USER'S OWN RISK. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR ANY CONTRIBUTOR OF INTELLECTUAL PROPERTY RIGHTS TO THE INTELLECTUAL PROPERTY BE LIABLE FOR ANY CLAIM, OR ANY DIRECT, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM ANY ALLEGED INFRINGEMENT OR ANY LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR UNDER ANY OTHER LEGAL THEORY, ARISING OUT OF OR IN CONNECTION WITH THE IMPLEMENTATION, USE, COMMERCIALIZATION OR PERFORMANCE OF THIS INTELLECTUAL PROPERTY.

This license is effective until terminated. You may terminate it at any time by destroying the Intellectual Property together with all copies in any form. The license will also terminate if you fail to comply with any term or condition of this Agreement. Except as provided in the following sentence, no such termination of this license shall require the termination of any third party end-user sublicense to the Intellectual Property which is in force as of the date of notice of such termination. In addition, should the Intellectual Property, or the operation of the Intellectual Property, infringe, or in LICENSOR's sole opinion be likely to infringe, any patent, copyright, trademark or other right of a third party, you agree that LICENSOR, in its sole discretion, may terminate this license without any compensation or liability to you, your licensees or any other party. You agree upon termination of any kind to destroy or cause to be destroyed the Intellectual Property together with all copies in any form, whether held by you or by any third party.

Except as contained in this notice, the name of LICENSOR or of any other holder of a copyright in all or part of the Intellectual Property shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Intellectual Property without prior written authorization of LICENSOR or such copyright holder. LICENSOR is and shall at all times be the sole entity that may authorize you or any third party to use certification marks, trademarks or other special designations to indicate compliance with any LICENSOR standards or specifications. This Agreement is governed by the laws of the Commonwealth of Massachusetts. The application to this Agreement of the United Nations Convention on Contracts for the International Sale of Goods is hereby expressly excluded. In the event any provision of this Agreement shall be deemed unenforceable, void or invalid, such provision shall be modified so as to make it valid and enforceable, and as so modified the entire Agreement shall remain in full force and effect. No decision, action or inaction by LICENSOR shall be construed to be a waiver of any rights or remedies available to it.

Contents

1. Scope 12

2. Conformance 15

3. References 18

3.1 Normative references 18

3.2 Other references 19

4. Terms and Definitions 20

5. Conventions 22

5.1 Abbreviated terms 22

5.2 UML notation 24

5.3 Used parts of other documents 24

5.4 Platform-neutral and platform-specific standards 24

5.5 Data dictionary tables 24

6. RESTful encoding of Sensor Planning Service for Earth Observation satellite Tasking (RESET) overview 26

6.1 RPC view of RESET 26

6.2 Resource view of RESET 28

6.3 Essential use cases 29

6.3.1 Planning with feasibility study 30

6.3.2 Direct planning 32

6.3.3 Reservation planning 34

7. Resource model 36

7.1 Resource root path 36

7.2 Capabilities resource 37

7.2.1 URL 37

7.2.2 Definition 38

7.2.3 HTTP methods 40

7.2.4 Error handling 40

7.3 Procedures resource 42
7.3.1 URL 42
7.3.2 Definition 42
7.3.3 HTTP methods 44
7.3.4 Error handling 45
7.4 Feasibility resource 46
7.4.1 URL 46
7.4.2 Definition 47
7.4.3 HTTP methods 49
7.4.4 Error handling 51
7.5 Planning resource 52
7.5.1 URL 52
7.5.2 Definition 53
7.5.3 HTTP methods 56
7.5.4 Error handling 58
7.6 Reservation resource 59
7.6.1 URL 59
7.6.2 Definition 60
7.6.3 HTTP methods 62
7.6.4 Error handling 63
8. RESET Core Requirement Class 64
8.1 Capabilities 65
8.1.1 Service Identification 65
8.1.2 Contents 66
8.1.3 Notifications 67
8.1.4 OperationsMetadata 68
8.1.5 URL resource distribution 69

8.2 Procedures 71
8.2.1 Tasking description 71
8.2.2 Sensor description 71
8.2.3 Sensor availabilities 72
8.3 Tasking 73
8.3.1 Task status 73
8.3.2 Task description 73
9. RESET Feasibility Requirement Class 74
9.1 Capabilities 74
9.2 Procedures 74
9.3 Feasibility 75
9.3.1 Creation 75
9.3.2 Task description 75
10. RESET Planning Requirement Class 76
10.1 Capabilities 76
10.2 Procedures 76
10.3 Planning 77
10.3.1 Status 77
10.3.2 Creation 78
10.3.3 Acquisition 79
10.3.4 Task description 79
10.3.5 Result description 79
10.3.6 Segments validation 80
11. RESET Feasibility Planning Requirement Class 81
11.1 Capabilities 81
11.2 Procedures 81
11.3 Feasibility 81

11.3.1 Submission 82
11.4 Planning 82
12. RESET Reservation Requirement Class 83
12.1 Capabilities 83
12.2 Procedures 83
12.3 Planning 83
12.4 Reservation 84
12.4.1 Creation 84
12.4.2 Modification 85
12.4.3 Submission 85
13. RESET Cancellation Requirement Class 87
13.1 Capabilities 87
13.2 Procedures 87
13.3 Tasking 87
13.3.1 Task cancellation 87
14. Annex A: Conformance Class Abstract Test Suite (Normative) 89
14.1 Conformance class: Core 90
14.1.1 Capabilities resource 90
14.1.2 Procedures resource 93
14.1.3 Tasking resources 95
14.2 Conformance class: Feasibility 96
14.2.1 Feasibility resource 96
14.3 Conformance class: Planning 99
14.3.1 Planning resource 99
14.4 Conformance class: Feasibility Planning 103
14.4.1 Feasibility resource 103
14.4.2 Planning resource 105

14.5 Conformance class: Reservation 108
14.5.1 Capabilities resource 108
14.5.2 Planning resource 108
14.5.3 Reservation resource 111
14.6 Conformance class: Cancellation 114
14.6.1 Capabilities resource 114
14.6.2 Planning resource 114
14.6.3 Reservation resource 115
15. Annex B: Revision history 117

Figures

Figure 2-1 RESET Requirement Classes 15
Figure 6-1 Planning with Feasibility study sequence diagram 31
Figure 6-2 Direct planning sequence diagram 33
Figure 6-3 Reservation planning sequence diagram 35
Figure 7-1 Operation Metadata example 38
Figure 7-2 Operation URL example 38

Tables

Table 2-1 RESET Requirement Classes 16
Table 5-1 Contents of data dictionary tables 25
Table 7-1 Capabilities URL 37
Table 7-2 Capabilities HTTP methods 40
Table 7-3 Capabilities Error Handling 41
Table 7-4 Procedures URL 42
Table 7-5 Procedures URL parameters 42
Table 7-6 Procedures HTTP methods 44
Table 7-7 Procedures HTTP methods parameters 44
Table 7-8 Procedures Error handling 45
Table 7-9 Feasibility URL 46
Table 7-10 Feasibility URL parameters 46
Table 7-11 Feasibility HTTP methods 49
Table 7-12 Feasibility HTTP methods parameters 50
Table 7-13 Feasibility Error handling 51
Table 7-14 Planning URL 52
Table 7-15 Planning URL parameters 52
Table 7-16 Planning: Results not available 54
Table 7-17 Planning: Segment cancellation 55
Table 7-18 Planning HTTP methods 56
Table 7-19 Planning HTTP methods parameters 57
Table 7-20 Planning Error handling 58
Table 7-21 Reservation URL 59
Table 7-22 Reservation URL parameters 59
Table 7-23 Reservation HTTP methods 62
Table 7-24 Reservation HTTP methods parameters 62
Table 7-25 Reservation Error handling 63
Table 8-1 Core operations 69
Table 8-2 Feasibility operations 69
Table 8-3 Planning operations 69
Table 8-4 Feasibility planning operations 69
Table 8-5 Reservation operations 70
Table 8-6 Cancellation operations 70
Table 10-1 Planning task status description 77
Table 10-2 Planning segments status description 77
Table 14-1 Conformance classes 89

Abstract

This **OGC**® Best Practices document specifies the interfaces, bindings, requirements and conformance classes that enable complete workflows for the tasking of sensor planning services for Earth Observation (EO) satellites. In fact it provides the interfaces for supporting the following **EO** sensor planning scenarios:

- Planning future acquisitions with feasibility study,
- Direct planning of future acquisitions,
- Reservation of planning for future acquisitions.

This specification includes a comprehensive list of sensor options and tasking options derived from the parent specification **OGC** 10-135 [NR22] which gathered inputs from several Satellite Agencies and Operators:

- ESA
- EUMETSAT
- CNES
- DLR
- CSA
- Airbus Defence & Space

This document is based on the standard:

> OGC 10-135, Sensor Planning Service Interface Standard 2.0 Earth Observation Satellite Tasking Extension, version 2.0. 2011.

which was initially produced during the ESA HMA (Heterogeneous Missions Accessibility) initiative [OR1] and related projects.

With respect to the parent specification this Best Practice document proposes the following changes:

- Replaces **SOAP** with **REST** for service encoding. This affects not only the way the service is implemented but also the way the standard is presented and described. In fact, basing the standard on **REST** implies that the service has to be described in terms of resources and methods applied on them whilst in SOAP services, the description is focusing on operations and in fact the **OGC** 10-135[NR22] is structured in Web Service operations.
- Usage of **OpenSearch** Description Documents as an alternate method for describing sensors and tasking Options (§7.3.2). This specification uses the sensors and tasking options model already described in the **OGC** 10-135 [NR22] standard but defines an additional method for describing sensors and tasking options within OpenSearch Description Documents based on the **OGC** 13-039 [NR23]. Actually this part of the specification refers to the **OpenSearch** Extension for Earth Observation Satellite Tasking.

Keywords

The following are keywords to be used by search engines and document catalogues.

ogcdoc, OGC document, sps, eo-sps, earth observation, eo, hma-s, sensor, planning, satellite, tasking, RESTful

Preface

This **OGC**® Best Practices document specifies the interfaces for issuing **EO** satellite tasking sensor planning in different scenarios (e.g. feasibility, planning, etc.) and with a lot of flexibility (feasibility options, planning options).

Due to the number of supported feasibility and planning scenarios covering different and also alternative needs, then a number of Requirements Classes have been defined collecting the specific requirements a conformant implementation has to comply with. In parallel, a number of Conformance Classes have been defined that regroup all the tests that a server has to pass in order to claim compliance with the corresponding Requirement Class. A server can comply with some of the Requirement Classes as it is not required to implement all classes.

Implementers shall be aware that:

- Not all scenarios (Requirement Classes) shall be implemented, only the Core one plus the others that are necessary for their use cases (see §0). However a server has to specify the supported Conformance Classes as evidence of the provided functionality.
- If tasking options are supported, then the implementation has to use a sub-set of the already identified scenarios unless they are not fitting with their needs; in that case an application profile listing the new scenarios to be modeled with SWE Common 2.0, has to be defined and implemented.

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 shall not be held responsible for identifying any or all such patent rights.

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

Submitting organizations

The following organizations submitted this Document to the Open Geospatial Consortium (OGC):

- Airbus Defence & Space
- European Space Agency
- Spacebel s.a.

Submitters

All questions regarding this submission should be directed to the editor or the submitters:

Name	Affiliation
Nicolas FANJEAU	Airbus Defence & Space
Sebastian ULRICH	Airbus Defence & Space

1. Scope

The scope of this Best Practice document is to describe the interfaces for providing an interoperable access to the tasking capabilities of various types of earth observation systems.

The interfaces can be used for determining the feasibility of an intended sensor planning request, for submitting such a request, for inquiring about the status of such a request, for updating or cancelling such a request and for requesting information on means of obtaining the data collected by the requested task.

It allows the preparation of different types of operation:

- GetCapabilities: describes abilities of the specific server implementation.
- DescribeSensor: describes a sensor.
- DescribeTasking: format of the tasking requests that can be submitted.
- GetSensorAvailibility / GetStationAvailibility: check the availability of a sensor or a ground station.
- GetFeasibility: perform the feasibility analysis process.
- Submit: submit a planning task described by the Tasking Parameters to the mission planning system.
- SubmitSegmentByID: submit a planning task created from the results of a Feasibility Analysis.

To verify whether the submission has been successfully completed and to retrieve / manage the results, dedicated operations have to be called:

- GetStatus: status of a planning task.
- GetTask: results of a Feasibility Study or of a submitted planning task.
- DescribeResultAccess: access to submitted products ready for download.
- Cancel: cancel of an already submitted planning task.
- Validate: manually validate an acquisition once it has been acquired.

A set of additional information permits the implementation of a reservation system. It includes the following operations:

- Reserve: reserve a task before submitting it.
- Update: change the parameters of a reserved task.
- Confirm: confirm the submission of a reservation and enter it to the mission planning system.

Tasking and Sensors Options are declared and set in two different formats:

- "Tasking and Sensors Options in Canonical XML format" (§8.1.4): the format defined in the parent Standard OGC 10-135 [NR22], represented by the eosps:TaskingParameters and swes:procedureDescriptionFormat elements which are complex structure regrouping options defined via SWE Common [NR20] and EO-SPS [NR22].
- "Tasking and Sensors Options in OpenSearch Description Document format" (§7.3.1 [NR23]): Tasking and sensors options can be declared in the OpenSearch Template URL similar to the other query parameters. However options elements in "Canonical XML format" are embedded as foreign mark-up in the OpenSearch Description Document in order to provide the information that a simple template parameter is not able to describe e.g. type, allowed values, default values etc.

This specification uses **REST** for defining its service interface: actually it is the re-engineering of the standard (replacing **SOAP** by **REST**):

> OGC 10-135, Sensor Planning Service Interface Standard 2.0 Earth Observation Satellite Tasking Extension, version 2.0. 2011.

Additionally to **REST** replacement of **SOAP**, a further modification has been done:

- Removed support for asynchronous notifications: in fact this document focuses on Web Applications in general and on **REST** approach in particular whilst the asynchronous message exchange is more of a **SOAP** Web Service Oriented function.

The REST encoding also affects the way the document is presented. In fact according to REST philosophy, the service is described in terms of Resources and Methods applied on them rather than in terms of operations as the parent specification did. From the Resource perspective, this Best Practice document defines the following:

- Service Metadata (§7.2): which is the resource document returning the description of the service capabilities, supported operations, supported collections and other service capabilities.
- Procedures (§7.3): which is the resource document returning the description of the sensors offered by the server.
- Ground Stations (§7.5): which is the resource document returning the description of the ground station.
- Feasibility (§7.4): which is the resource to perform a Feasibility Analysis on a given sensor or type of sensor (optical, SAR, etc.), as well as accessing its results.
- Reservation (§7.6): which is the resource allowing a user to reserve a task before submitting it.
- Planning (§7.5): which is the resource allowing a user to submit a planning task, access its status, its results, as well as cancelling a task.

For each resource, the document specifies:

- The definition, via XML schema, of the various items composing the resource.
- The HTTP Methods available (according to the **REST** – Uniform Interface concept) e.g. **GET**, **POST**, **PUT** and **DELETE**.
- The HTTP URLs for accessing the resource.
- The connections with the other resources (according to the REST – Connectedness concept).

The section §6 provides a more detailed description of the **REST** approach for this specification and the mapping with the parent Standard document.

Due to the flexibility of the protocol, several conformance classes have been defined, each addressing a specific use case. A compliant implementation does not have to comply with all of these classes but with at least Core class. Compliance to the remaining classes is optional.

2. Conformance

This Best Practice document defines the interfaces and the requirements for implementing a server supporting the feasibility and planning for Earth Observation satellite sensors.

Therefore different Requirement Classes have been defined specifying the requirements for implementing the feasibility and planning processes. Moreover additional classes have been considered for defining optional functions that might be implemented by servers having extended functionality.

The complete list of Requirements Classes is listed below:

- **Core**, which specifies the minimum behaviour that all **RESET** servers shall implement. It includes the support of the operations Tasking description, Sensor description or Task description.
- **Feasibility**, which specifies the additional requirements a **RESET** server has to implement for performing a feasibility analysis process.
- **Planning**, which specifies the additional requirements a **RESET** server shall implement for performing a planning process.
- **Feasibility Planning**, which specifies the additional requirements a **RESET** server shall implement for performing a feasibility analysis process and then submitting the subsequent results.
- **Reservation**, which specifies the additional requirements a **RESET** server shall implement for performing a feasibility analysis process and then submitting the subsequent results.
- **Cancellation**, which specifies the additional requirements a **RESET** server shall implement for performing the cancellation of planning or reservation processes.

The following diagram shows the relationships between the defined Requirement Classes.

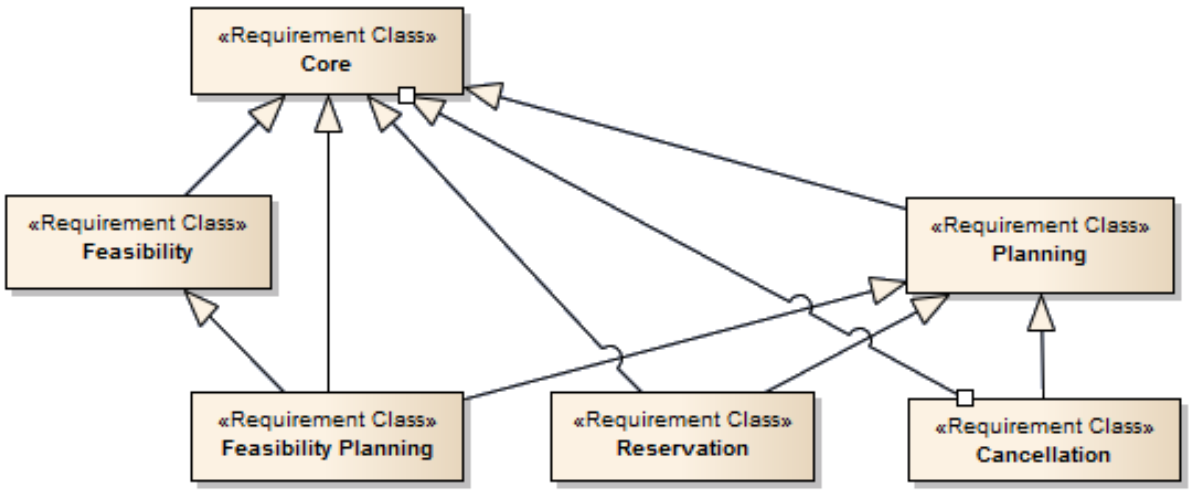

Figure 2-1 RESET Requirement Classes

The inheritance relationship between the different classes represents the inheritance of all requirements from the super class. E.g.: The Feasibility class defines its specific requirements and includes the requirements defined in the Core class. Hence a Server claiming compliance to the Feasibility class has to comply with all the Feasibility and Core class requirements.

The following table reports:

- The Requirement Class name,
- The URI,
- The dependency with other requirements classes.

Table 2-1 RESET Requirement Classes

Requirement Class	Requirement Class URI	Dependency
Core	http://www.opengis.net/spec/RESET/1.0/req/Core	
Feasibility	http://www.opengis.net/spec/RESET/1.0/req/Feasibility	Core
Planning	http://www.opengis.net/spec/RESET/1.0/req/Planning	Core
Feasibility Planning	http://www.opengis.net/spec/RESET/1.0/req/FeasibilityPlanning	Core Feasibility Planning
Reservation	http://www.opengis.net/spec/RESET/1.0/req/Reservation	Core Planning
Cancellation	http://www.opengis.net/spec/RESET/1.0/req/Cancellation	Core Planning

The root path of all Requirements and conformance test URIs defined in this document is:

http://www.opengis.net/spec/RESET/1.0/

Conformance with this Best Practice document shall be checked using all the relevant tests specified in Annex A: Conformance Class Abstract Test Suite (Normative) of this document. The framework, concepts, and methodology for testing as well as the criteria to be achieved in order to claim conformance are specified in the **OGC** Compliance Testing Policies and Procedures and the **OGC** Compliance Testing web site1.

1 www.opengeospatial.org/cite

In order to conform to this **OGC**® interface standard, a software implementation shall choose to implement at least Core class and optionally the other classes.

A server that complies only with the Core class is a server supporting feasibility and planning based on identified EO sensors.

All requirements-classes and conformance-classes described in this document are owned by the standard(s) identified.

3. 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.

3.1 Normative references

[NR1] IETF RFC4287, ATOM The Atom Syndication Format. 2005

[NR2] IETF RFC 3066, Tags for the Identification of Languages. 2001

[NR3] IETF RFC 3339, Date and Time on the Internet: Timestamps. 2002

[NR4] IETF RFC 5988, Web Linking. 2010

[NR5] OpenSearch, OpenSearch Specification, version 1.1 draft 5.

[NR6] OpenSearch, OpenSearch Geo Extension, version 1.0 draft 2.

[NR7] OpenSearch, OpenSearch Time Extension, version 1.0 draft 1.

[NR8] OpenSearch, OpenSearch Parameters Extension, version 1.0 draft 2.

[NR9] OASIS, SearchRetrieve: Part 4. APD Binding for OpenSearch, version 1.0. 2013.

[NR10] OGC 10-176r4, New OGC Document Template Draft.

[NR11] OGC 10-177r2, Guidance to Editors and Authors of OGC Documents that use the OGC Standards document template, version 1.2. 2012.

[NR12] OGC 06-135r11, Policy Directives for Writing and Publishing OGC Standards: TC Decisions. 2011.

[NR13] OGC 08-131r3, The Specification Model - A Standard for Modular specifications, version 1.0. 2009.

[NR14] OGC 05-008, OGC Web Services Common Specification, version 1.0. 2005.

[NR15] OGC 06-121r9, OWS Common Implementation Specification, version 2.0. 2010.

[NR16] OGC 06-103r4, OGC® Implementation Standard for Geographic information – Simple feature access - Part 1: Common architecture, version 1.2.1. 2011.

[NR17] OGC 10-157r4, Earth Observation Metadata profile of Observations & Measurements, draft 1 2014.

[NR18] OGC 10-026r2, OpenSearch Earth Observation product Extension. 2013.

[NR19] OGC 10-032r6, OpenSearch GeoSpatial and Temporal Extensions. 2013.

[NR20] OGC 08-094r1, SWE Common Data Model Encoding Standard, version 2.0.0. 2011.

[NR21] OGC 09-000, Sensor Planning Service Implementation Standard, version 2.0. 2011.

[NR22] OGC 10-135, Sensor Planning Service Interface Standard 2.0 Earth Observation Satellite Tasking Extension, version 2.0. 2011.

[NR23] OGC 10-039, OpenSearch Extension for Earth Observation Satellite Tasking, version 1.0. 2013.

[NR24] OGC 09-001, OGC SWE Service Model Implementation Standard, version 2.0. 2011.

[NR25] OGC 13-042, RESTful Encoding of Ordering Services Framework For Earth Observation Products, version 1.0. 2014.

3.2 Other references

[OR1] ESA, TM-21, Design Methodology, Architecture and Use of Geospatial Standards for the Ground Segment Support of Earth Observation Missions. 2012.

4. Terms and Definitions

This document uses the terms defined in Sub-clause 5.3 of [OGC 06-121r8], which is based on the ISO/IEC Directives, Part 2, Rules for the structure and drafting of International Standards. In particular, the word “shall” (not “must”) is the verb form used to indicate a requirement to be strictly followed to conform to this standard.

For the purposes of this document, the following additional terms and definitions apply.

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

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

OpenSearch
Draft specification for web search syndication, originating from Amazon's A9 project and given a corresponding interface binding by the OASIS Search Web Services working group.

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

procedure oriented architectural style
Platform-independent design approach that is focused on operations, their parameters and their results, that can be defined in an abstract level specification. Concrete platform dependent specifications can be derived from the abstract level, allowing, for example, KVP or SOAP messaging.

requirement
Something that is necessary in advance.

resource oriented architectural style
Platform-independent design approach that is focused on resources, representations and actions, that can be defined in an abstract level specification. Concrete platform dependent specifications can be derived from the abstract level, allowing, for example, a RESTful architecture.

state
Condition that persists for a period.

Task
Conceptual resource that represents an SPS assignment. It includes the (possibly empty) set of tasking parameters.

Tasking
Parameterizing an asset; can be done by sending one or more tasking requests.

Tasking request
Request with certain tasking semantics that contains tasking parameters.

Tasking Parameter
Parameter that has an influence on the parameterization of an asset.

URL component
Text contained after the context path between two forward slashes "/", that are not placed between two matching parentheses "(...)", and before any question mark character "?".

URL component key
Text contained in a URL component that comes before an opening parenthesis "(".

URL component value
Text contained in a URL component that is enclosed between the first opening parenthesis "(" and the last closing parenthesis ")" in the → component.

Example In the http://example.org/component1/componentKey(componentValue), components are as follows:

Context Path: http://example.org/
URL Components: component1, componentKey(componentValue)
URL Component Keys: component1, componentKey
URL Component Values:componentValue

5. Conventions

This sections provides details and examples for any conventions used in the document. Examples of conventions are symbols, abbreviations, use of XML schema, or special notes regarding how to read the document.

5.1 Abbreviated terms

Some of the abbreviated terms listed in sub clause 5.1 of the OWS Common Implementation Specification (OGC 06-121) apply to this document, plus the following abbreviated terms.

API	Application Program Interface
ATM	Atmospheric
COTS	Commercial Off The Shelf
CQL	Common Query Language
CRS	Coordinate Reference System
DCE	Distributed Computing Environment
DCP	Distributed Computing Platform
DDS	Data Dissemination Service
EC	European Commission
EO	Earth Observation
EO-SPS	Earth Observation Satellite Tasking Extension for the SPS 2.0 Standard
ESA	European Space Agency
GCM	GMES Collaborative Mission
GMES	Global Monitoring for Environment and Security
GML	Geographic Markup Language
HMA	Heterogeneous Mission Accessibility
HMA-AWG	HMA Architecture Working Group
HTTP	Hypertext Transport Protocol
IETF	Internet Engineering Task Force
ISO	International Organisation for Standardisation
LTDP	Long Term Data Preservation

O&M	Observations & Measurements
OGC	Open Geospatial Consortium
OSDD	OpenSearch Description Document
OASIS	Organisation for the Advancement of Structured Information Standards
OPT	Optical
RESET	RESTful encoding of Sensor Planning Service for Earth Observation satellite Tasking
REST	Representational State Transfer
ROSEO	RESTful Encoding of Ordering Services Framework for Earth Observation Products
RSS	Rich Site Summary
SAR	Synthetic Aperture Radar
SensorML	Sensor Model Language
SPS	Sensor Planning Service
SWE	Sensor Web Enablement
SWS	Search Web Services
UML	Unified Modelling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URL	Uniform Resource Name
UUID	Universal Unique Identifier
URN	Uniform Resource Name
UTF-8	Unicode Transformation Format-8
WSDL	Web Service Definition Language
W3C	World Wide Web Consortium
WKT	Well Known Text
XML	eXtensible Markup Language

5.2 UML notation

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

5.3 Used parts of other documents

This document uses significant parts of document OGC 10-032. 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%).

5.4 Platform-neutral and platform-specific standards

As specified in Clause 10 of OGC Abstract Specification Topic 12 "OGC Service Architecture" (which contains ISO 19119), this document includes both Distributed Computing Platform-neutral and platform-specific standards. This document first specifies each operation request and response in platform-neutral fashion. This is done using a table for each data structure, which lists and defines the parameters and other data structures contained.

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).

5.5 Data dictionary tables

When the data type used for this parameter, in the third column of such a table, is an enumeration or code list, all the values specified by a specific OWS shall be listed, together with the meaning of each value. When this information is extensive, these values and meanings should be specified in a separate table that is referenced in the third column of this table row.

The data type of many parameters, in the third table column, 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 contents of these data dictionary tables are normative, including any table footnotes.

The UML model data dictionary is specified herein in a series of tables.

Table 5-1 Contents of data dictionary tables

Column title	Column contents
Names (left column)	Two names for each included parameter or association (or data structure). The first name is the UML model attribute or association role name. The second name uses the XML encoding capitalization specified in sub-clause 11.6.2 of [OGC 05-008]. The name capitalization rules used are specified in sub-clause 11.6.2 of [OGC 05-008]. Some names in the tables may appear to contain spaces, but no names contain spaces.
Definition (second column)	Specifies the definition of this parameter (omitting un-necessary words such as "a", "the", and "is"). If the parameter value is the identifier of something, not a description or definition, the definition of this parameter should read something like "Identifier of TBD".
Data type and value (third column) or Data type (if are no second items are included in rows of table)	Normally contains two items: The mandatory first item is often the data type used for this parameter, using data types appropriate in a UML model, in which this parameter is a named attribute of a UML class. Alternately, the first item can identify the data structure (or class) referenced by this association, and references a separate table used to specify the contents of that class (or data structure). The optional second item in the third column of each table should indicate the source of values for this parameter, the alternative values, or other value information, unless the values are quite clear from other listed information.
Multiplicity and use (right or fourth column) or Multiplicity (if are no second items are included in rows of table)	Normally contains two items: The mandatory first item specifies the multiplicity and optionality of this parameter in this data structure, either "One (mandatory)", "One or more (mandatory)", "Zero or one (optional)", or "Zero or more (optional)". The second item in the right column of each table should specify how any multiplicity other than "One (mandatory)" shall be used. If that parameter is optional, under what condition(s) shall that parameter be included or not included? If that parameter can be repeated, for what is that parameter repeated?

6. RESTful encoding of Sensor Planning Service for Earth Observation satellite Tasking (RESET) overview

As highlighted in the introduction sections, this document provides a simplified and **RESTful** encoding of the parent specification Earth Observation Satellite Tasking Extension for **SPS** 2.0 [NR22].

This is a **RESTful** specification and so it is described in terms of resources. However it is derived from [NR22], which is a SOAP Web Service and thus is both operations centric and following an "RPC style" Both techniques have almost the same functionality. This then means that this specification can be identically described from the two different points of view. This section summarizes this specification from both a Resource point of view and from an **RPC** style point of view and explains how they relate to each other.

6.1 RPC view of RESET

RESET deals with the following information items:

- **Sensor description**, provides descriptions of the sensors that can be tasked. These descriptions can be provided in multiple **OGC** standards, including SensorML, SOS, etc.
- **Tasking requests**, are the requests to be sent. These contain all the information to configure the acquisition of a sensor, or set of sensors;
- **Feasibility Study**, describes the results of a feasibility request. This preliminary process allows the calculation of the possibilities offered by a given sensor to perform a given tasking request. This structure contains a list of **Segments**, which represent a given acquisition.
- **Programming Status**, describes the results of the submission of a task. It describes the current status of the **Segment** which represents a single acquisition.

From RPC perspective, this specification is composed of the following operations:

- **GetCapabilities**: allows a client to request and receive a service metadata (or Capabilities) document that describes the abilities of the specific server implementation.
- **DescribeSensor**: allows a client to request the complete description of a sensor, including the modes it can perform, choosing between the various descriptions formats described in the GetCapabilities.
- **DescribeTasking**: this operation allows a client to get the format of the tasking requests that can be submitted to the server. This includes :
    - Area Of Interest (AOI): the area of the globe to be tasked. Extra information includes the kind of AOI (polygon, circle, etc.), the format of the coordinates, etc.
    - Time Of Interest (TOI): the periods that are to be tasked, with the format of these dates (e.g. ISO 8601).
    - Acquisition type : additional information that includes instrument (sensor) modes, acquisition angles, validation parameters such as cloud or snow coverage, …

- **GetSensorAvailability** / **GetStationAvailability**: these operations allow a user to check the availability of a sensor or a ground station.
- **GetFeasibility**: this operation requests the feasibility analysis process to be performed. The Segments provided by this function can then be submitted by the SubmitSegmentByID operation. Its results are also available through the GetTask operation.
- **Submit**, allows the submission of a planning task described by the Tasking Parameters to the mission planning system.
- **SubmitSegmentByID**, allows a user to submit a planning task created from the results of a Feasibility Analysis.

  These operations are "asynchronous" meaning that they perform the submission of the task to the mission planning system and return a summary containing the status of the Segments that have been submitted.

  To verify whether the submission has been successfully completed and to retrieve / manage the results, dedicated operations have to be called.

- **GetStatus**, allows the retrieval of the status of a planning task.
- **GetTask**, allows the retrieval of either the results of a Feasibility Study or of a submitted planning task.
- **DescribeResultAccess**, allows access to the submitted products ready for download once they have been acquired, downloaded to the Ground Station and processed as required.
- **Cancel** cancels a previously submitted planning task. This operation is a two-part process : the first call will evaluate the possibility of the cancellation and return a status report showing which segments can, or cannot, be cancelled, along with the time this report will be valid. The second call will then validate the cancellation of the segments that are cancellable.
- **Validate** allows the user to manually validate an acquisition once it has been acquired. Indeed, several attempts are sometimes needed for earth observation systems to acquire satisfactory data for an area of the earth surface. For instance, optical acquisitions are subject to cloud cover that can prevent clear imaging of the region of interest. In this case new acquisitions are performed until the region of interest is imaged properly. This operation provides a means to confirm that the last acquisition fits the user's needs.
- A set of additional information that allows the implementation of a reservation system. It includes the following operations :
    - **Reserve** that allows the user to reserve a task before submitting it, for example if the user wants to perform another feasibility study before confirming.
    - **Update** allows the changing of the parameters of a reserved task.
    - **Confirm** allows a user to confirm the submission of a reservation and enter it to the mission planning system.

6.2 Resource view of RESET

From the "Resource" perspective, this specification is composed of the following resources:

- **Service Metadata**: Describes the service capabilities, supported operations, supported collections and other service capabilities.
  The following HTTP methods are used:
  - **GET**: returns the capabilities of the system.
- **Procedures**: Describes the sensors offered by the server.
  The following HTTP methods are used:
  - **GET /procedures/{sps:procedure}/{swes: procedureDescriptionFormat?}**: equivalent to the DescribeSensor operation. Allows the retrieval of the capabilities of a sensor in the given format (providing it is used by the server).
  - **GET /procedures/{sps:procedure}/tasking**: equivalent to DescribeTasking.
  - **GET /procedures/{sps:procedure}/availability**: equivalent to GetSensorAvailibility.
- **Ground Stations**: Retrieves information about a ground station.
  - **GET /groundstation/{eosps:station}**: Describes the ground station.
  - **GET /groundstation/{eosps:station}/availability**: equivalent to GetStationAvailability.
- **Feasibility**: Perform a Feasibility Analysis on a given sensor or type of sensor (optical, SAR, etc.), as well as accessing its results.
  The following HTTP methods are used :
  - **POST /feasibility**: equivalent to GetFeasibility. The content of the request shall be based on the response provided by the DescribeTasking operation.
  - **POST /feasibility/{sps:taskID}**:equivalent to SubmitSegmentByID in the RPC view of RESET 6.1 above. The method submits a set of segments that have been produced by a Feasibility Analysis.
  - **GET /feasibility/{ sps:taskID}**: equivalent to GetTask. The method retrieves the results of a Feasibility Analysis.
- **Reservation**: allows the user to reserve a task before submitting it.
  The following HTTP methods are used :
  - **POST /reservation**: equivalent to the Reserve operation. Allows the user to reserve a planning task for a certain time.
  - **POST /reservation/{ sps:taskID}**: equivalent to the Confirm operation. Allows the user to submit a planning task that was previously reserved.
  - **PUT /reservation{sps:taskID}**: equivalent to the Update operation. Allows the user to apply some modifications to a previously reserved task.
- **Planning**: allows the user to submit a planning task, accessing its status, its results, as well as cancelling a task.
  The following HTTP methods are used:
  - **POST /planning**: equivalent to the Submit operation in the RPC view of RESET. The method provides a means to submit a planning task. The content of the

request shall be based on the response provided by the DescribeTasking operation.

- **GET /planning/{ sps:taskID}**: equivalent to the GetStatus operation. Checks the status of a task.
- **GET /planning/{ sps:taskID}/segments**: equivalent to the GetTask operation. Retrieves a complete report of the task, with description of all segments submitted and their status. It should also be used prior to a Cancel request in order to assess the possibility of the cancellation.
- **POST /planning/{ sps:taskID}/segments/{ eosps:segmentID}**: equivalent to the Validate operation. Allows the user to validate an acquisition as corresponding to the user's needs.
- **GET /planning/{ sps:taskID}/results**: equivalent to DescribeResultAccess.
- **DELETE /planning/{ sps:taskID}**: equivalent to the Cancel operation. Cancels a selected task.

6.3 Essential use cases

The scenarios described here demonstrate the three basic use cases that perform the submission of a planning task to a (set of) mission planning system(s).

In these scenarios, the following entities are specified:

- **RESET** client: the client application or user that will send the requests to the client.
- **RESET** server: the server application that will handle those requests.
- Download Server: the server location where the user will be able to download the results of the process.

6.3.1 Planning with feasibility study

The typical scenario is:

- The client gets the capabilities of the server using the GetCapabilities operation (**GET /**).
- Selects the sensor to be tasked from the GetCapabilities and optionally (if more information is required) the DescribeSensor methods (**GET /procedures/{sps:procedure}/{swes:sensorDescriptionFormat?}**). The user then requests the format of the Tasking parameters needed to submit a task, through the DescribeTasking operation (**GET /procedures/{sps:procedure}/tasking**).
- Then this structure is filled in on the client side, allowing the user to choose the area of interest, the time of interest, as well as other parameters such as the instrument mode.
- This structure is then sent to the server as part of a GetFeasibility request (corresponding to **POST /feasibility**). This will return a set of segments representing acquisitions that fulfill the given tasking parameters.
- The user will then assess these results in order to choose the segments required to be submitted for acquisition. These segments will then be submitted using the SubmitSegmentByID (**POST /feasibility/{taskID}**). This will start a new planning task.
- After that, the following events are possible:
    - The user requests the status of the task using the GetStatus operation (**GET /planning/{taskID}**), or the complete description of the task using the GetTask operation (**GET /planning/{taskID}/segments**).
    - The user cancels the task using the Cancel operation (**DELETE /planning/{taskID}**).
- Once the task is finished, which corresponds to all segments involved begin acquired, the user can do the following:
    - Get the data describing a given segment, through the GetTask (**GET /planning/{taskID}/segments**), and the results with the DescribeResultAccess operation (**GET /planning/{taskID}/results**). This will provide the means to connect to the Download Server in order to get the acquired products.
- After assessing these results, validate a given segment using the Validate (**POST /planning/{taskID}/segments/{segmentID}**) operation.

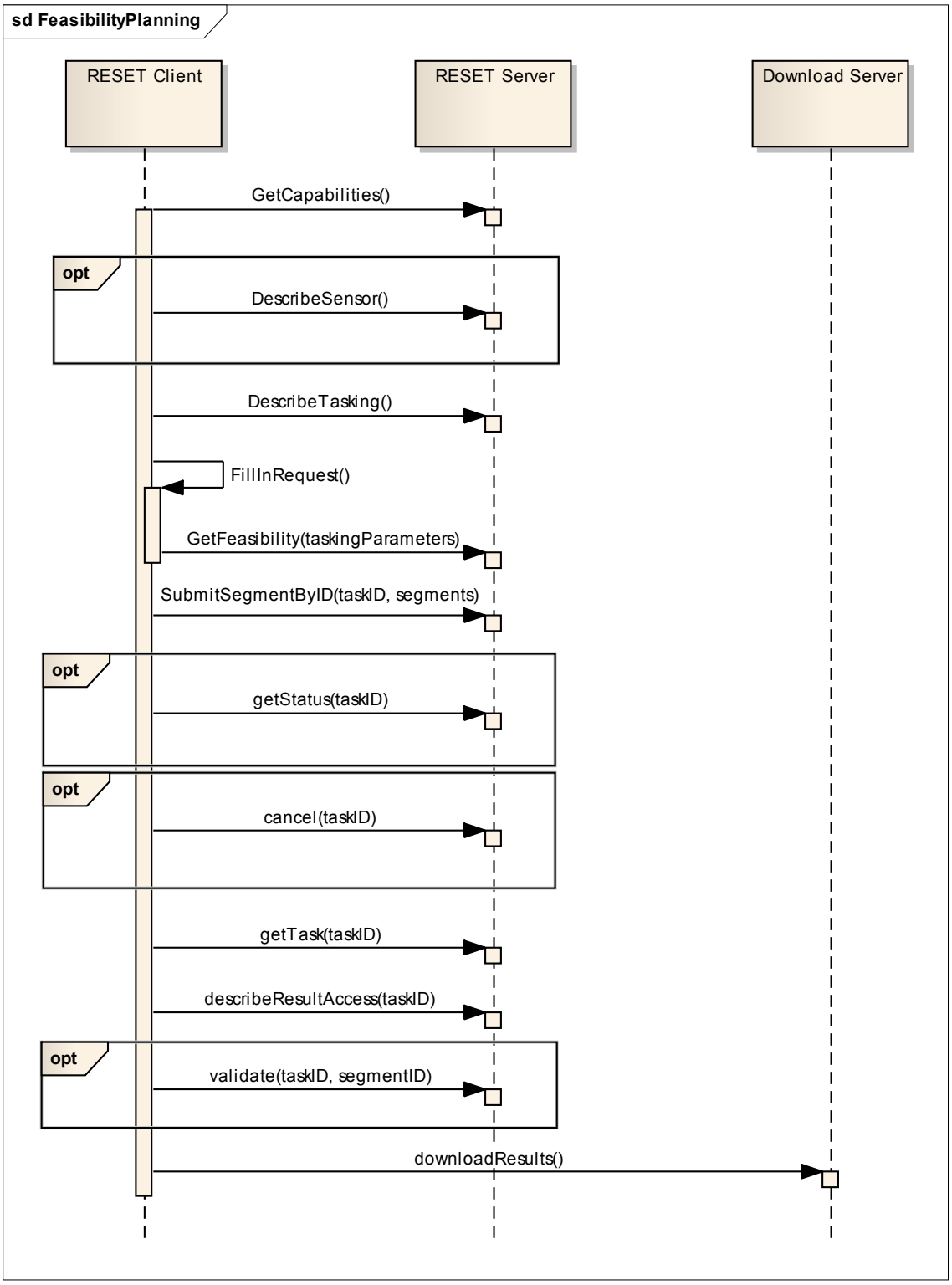

Figure 6-1 Planning with Feasibility study sequence diagram

6.3.2 Direct planning

The typical scenario is:

- The client gets the capabilities of the server using the GetCapabilities operation (**GET /**).
- Selects the sensor to be tasked from the GetCapabilities and optionally (if more information is required) the DescribeSensor methods (**GET /procedures/{sps:procedure}/{swes:sensorDescriptionFormat?}**). The user then requests the format of the Tasking parameters needed to submit a task, through the DescribeTasking operation (**GET /procedures/{sps:procedure}/tasking**).
- Then this structure is filled in on the client side, allowing the user to choose the area of interest, the time of interest, as well as other parameters such as the instrument mode.
- This structure is then sent to the server as part of a Submit request (corresponding to **POST /planning**). This will return a set of segments representing the acquisitions that have been submitted to fulfill the given request. This will start a new planning task.
- After that, the following events are possible:
    - The user requests the status of the task using the GetStatus operation (**GET /planning/{taskID}**), or the complete description of the task using the GetTask operation (**GET /planning/{taskID}/segments**).
    - The user cancels the task using the Cancel operation (**DELETE /planning/{taskID}**).
- Once the task is finished, which corresponds to all segments involved begin acquired, the user can do the following:
    - Get the data describing a given segment, through the GetTask (**GET /planning/{taskID}/segments**), and the results with the DescribeResultAccess operation (**GET /planning/{taskID}/results**). This will provide the means to connect to the Download Server in order to get the acquired products.
- After assessing these results, validate a given segment using the Validate (**POST /planning/{taskID}/segments/{segmentID}**) operation.

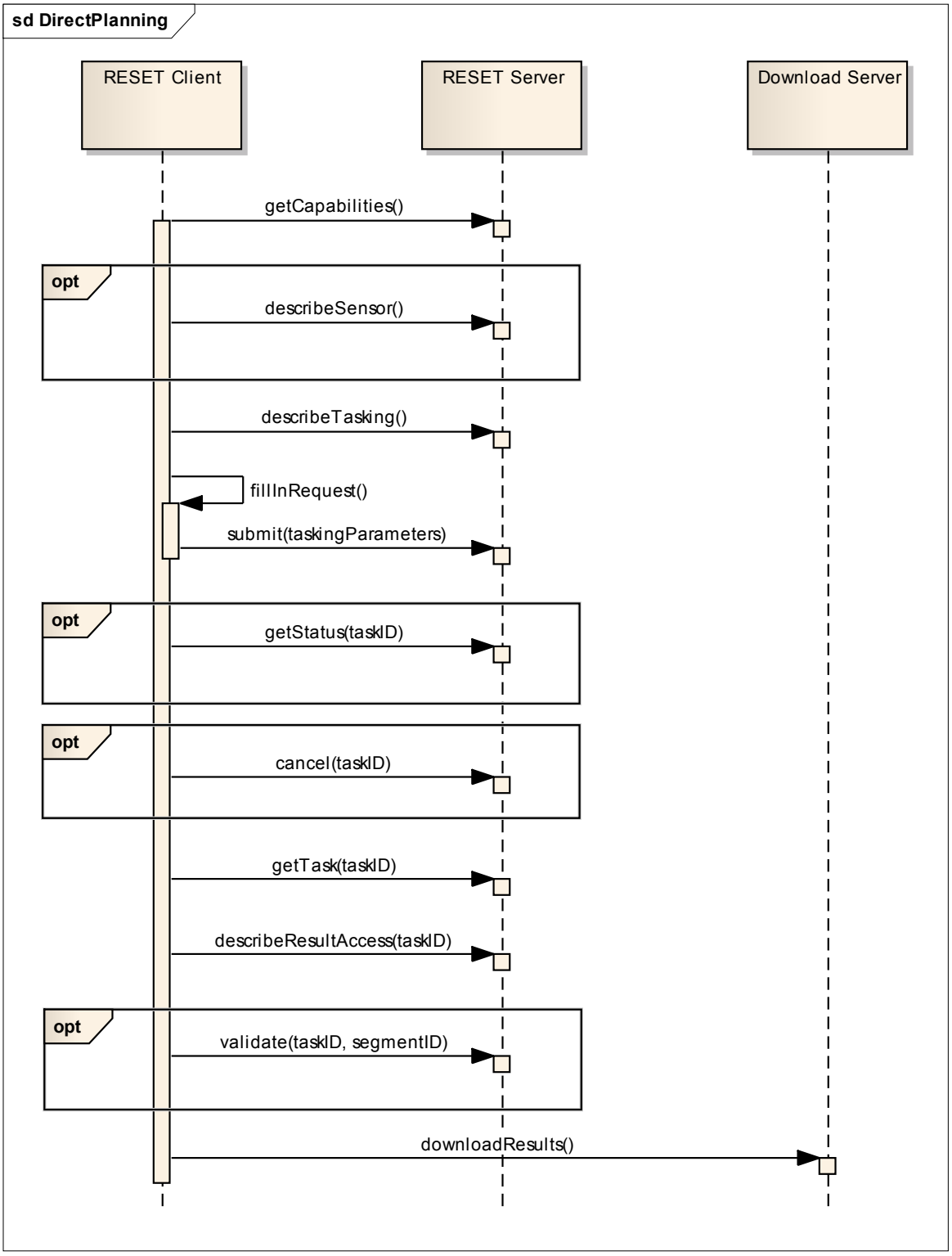

Figure 6-2 Direct planning sequence diagram

6.3.3 Reservation planning

The typical scenario is:

- The client gets the capabilities of the server using the GetCapabilities operation (**GET /**).
- Selects the sensor to be tasked from the GetCapabilities and optionally (if more information is required) the DescribeSensor methods (**GET /procedures/{sps:procedure}/{swes:sensorDescriptionFormat?}**). The user then requests the format of the Tasking parameters needed to submit a task, through the DescribeTasking operation (**GET /procedures/{sps:procedure}/tasking**).
- Then this structure is filled in on the client side, allowing the user to choose the area of interest, the time of interest, as well as other parameters such as the instrument mode.
- This structure is then sent to the server as part of a Reserve request (corresponding to **POST /reservation**). This will return a set of segments representing the acquisitions that have been submitted to fulfill the given request. This will start a new reservation task.
- This will reserve the task. The user then has the possibility, until the reservation expiration time, to change the request by using the Update operation (PUT /reservation/{taskID}).
- If the result is satisfactory, the user can then submit the task with the Confirm operation (POST /reservation/{taskID}). This will start a new planning task.
- After that, the following events are possible:
    - The user requests the status of the task using the GetStatus operation (**GET /planning/{taskID}**), or the complete description of the task using the GetTask operation (**GET /planning/{taskID}/segments**).
    - The user cancels the task using the Cancel operation (**DELETE /planning/{taskID}**).
- Once the task is finished, which corresponds to all segments involved begin acquired, the user can do the following:
    - Get the data describing a given segment, through the GetTask (**GET /planning/{taskID}/segments**), and the results with the DescribeResultAccess operation(**GET /planning/{taskID}/results**). This will provide the means to connect to the Download Server in order to get the acquired products.
- After assessing these results, validate a given segment using the Validate (**POST /planning/{taskID}/segments/{segmentID}**) operation.

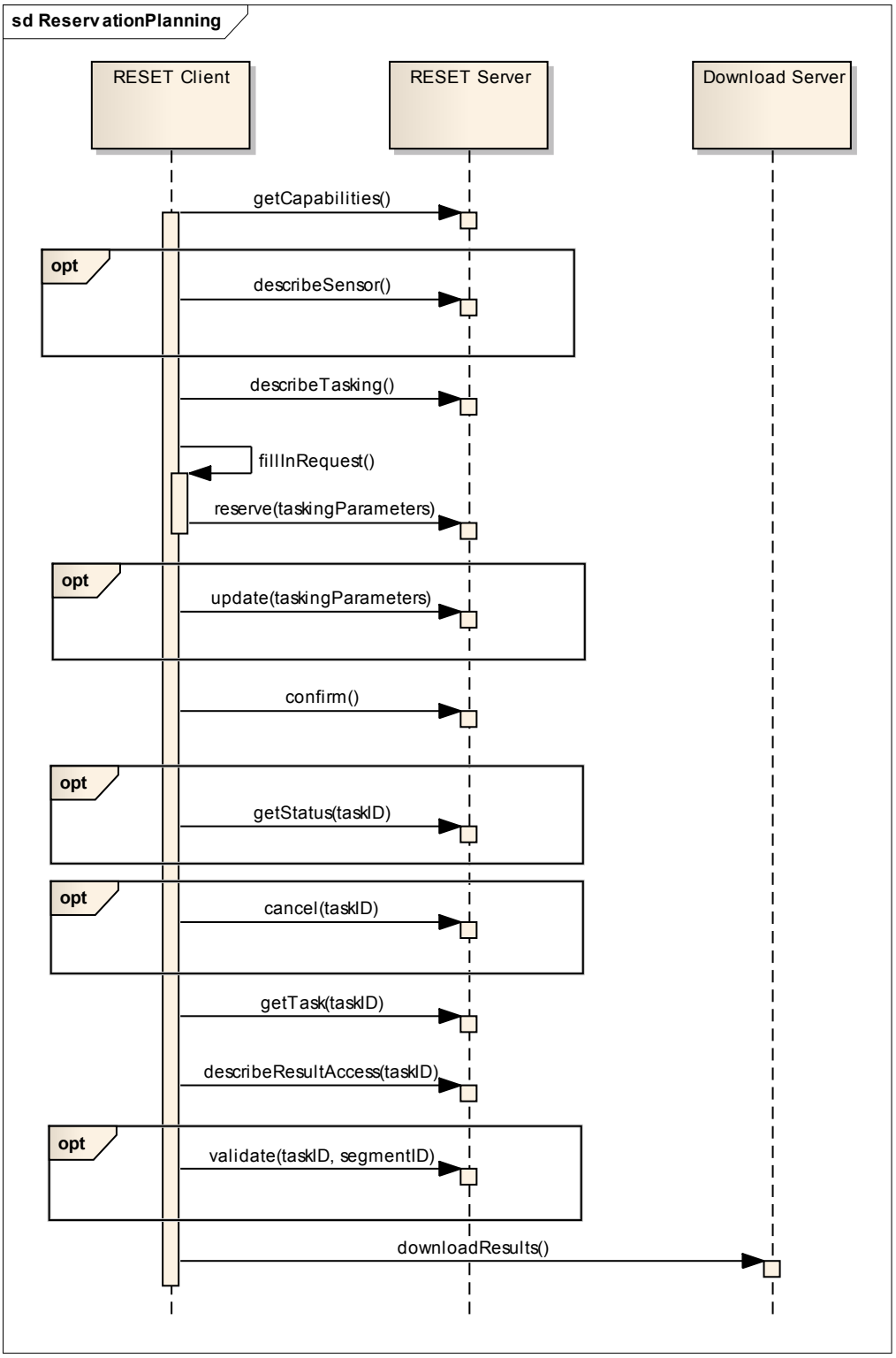

Figure 6-3 Reservation planning sequence diagram

7. Resource model

This section reports all resources defined in this specification including:

- The **URL** addressing the resource;
- The detailed definition, via XML schema notations;
- Supported HTTP methods (e.g. GET, POST) and provided operations.

A sub-section is dedicated to each of them.

7.1 Resource root path

All RESET resources belong to the same root path defined as follow:

- <serviceRootURL>: http://<hostName>:<hostPort>/<contextPath>
- <ResetRootPath>: <serviceRootURL>/RESET/<version>

Where:

- <hostName>: the IP address of the server, or the associated DNS address.
- <hostPort>: the port at which the application server containing the server listens.
- <contextPath>: the path of the web application implementing the **RESET** interface, within the application server context.
- <version>: the version of the **RESET** standard implemented by the server, here 1.0.

7.2 Capabilities resource

Capabilities resource contains metadata about the **RESET** server's capabilities, e.g. identifier, description, restrictions, supported operations, contents, etc.

These capabilities are based on the **SPS** Capabilities structure, which is itself based on the **OWS** Capabilities structure, applying the HTTP profile. This profile describes URIs to access with given HTTP methods to a **RESTful** web service.

Therefore, it contains links to all the other resources accessible on the server, by describing the URLs that provide access to these resources.

7.2.1 URL

Table 7-1 Capabilities URL

URL	Definition
<CapabilitiesResourceURL>	<ResetRootPath>
<ServiceIdentification>	<CapabilitiesResourceURL>/identification
<ServiceProvider>	<CapabilitiesResourceURL>/provider
<OperationsMetadata>	<CapabilitiesResourceURL>/operations
<Notifications>	<CapabilitiesResourceURL>/notifications
<Contents>	<CapabilitiesResourceURL>/contents

7.2.2 Definition

The capabilities document is described in the SPS Specification [NR21]. Therefore, the only part described here will be the OperationMetadata, which will create the URIs to access the other resources.

This structure used is the following:

```
<ows:OperationsMetadata xmlns:ows="http://www.opengis.net/ows/1.1">
        <ows:Operation>
        <ows:DCP>
        <ows:HTTP>
                <ows:Get
xmlns:xlin="http://www.w3.org/1999/xlink"
xlin:href=" <ResetRootPath>/procedures/{procedure}/tasking">
                <ows:Constraint name="procedure">
                        <ows:AllowedValues>
                                <ows:Value>OPT</ows:Value>
                        </ows:AllowedValues>
                        <ows:AllowedValues>
                                <ows:Value>SAR</ows:Value>
                        </ows:AllowedValues>
                </ows:Constraint>
                </ows:Get>
        </ows:HTTP>
        </ows:DCP>
        </ows:Operation>
<ows:OperationsMetadata>
```

Figure 7-1 Operation Metadata example

In this example, the operation described is the tasking request description (DescribeTasking operation). The URI to task is the one described in the "href" parameter of the Get element, which is the HTTP method to be used. Within the href parameter, the words between brackets, e.g. {procedure}, are parameters to be filled in when submitting a request. For example, from this description, the URI to be constructed would be:

```
<ResetRootPath>/procedures/OPT/tasking
```

Figure 7-2 Operation URL example

However the **OWS** specification did not foresee the use of the **PUT** and **DELETE** HTTP methods. Neither did it provide a way to add these to the OperationsMetadata structure without redefining the whole structure. Therefore the OperationsMetadata structure was replaced in the **ROSEO** specification [NR24]by the following:

```
<ns:resourceURL resource="feasibility" methods="POST"
URL="http://127.0.0.1:8080/DreamServices/dream/mmfas/gmes-mmfas/
        reset/1.0.0/feasibility/"/>
                <ns:resourceURL resource="feasibility" methods="GET"
URL="http://127.0.0.1:8080/DreamServices/dream/mmfas/gmes-mmfas/
        reset/1.0.0/feasibility/{taskID}"/>
                <ns:resourceURL resource="feasibility" methods="GET"
URL="http://127.0.0.1:8080/DreamServices/dream/mmfas/gmes-mmfas/
        reset/1.0.0/feasibility/{taskID}/segments/"/>
```

7.2.3 HTTP methods

Table 7-2 Capabilities HTTP methods

HTTP Method	URL	Definition	Request Entity Body	Response Entity Body
GET	<RESET Root Path>	It returns the complete capabilities document.	Empty	<sps:Capabilities> element instance
	<RESET Root Path>/ServiceIdentification	It returns the ServiceIdentification element.	Empty	<ows:ServiceIdentification> element instance
	<RESET Root Path>/ServiceProvider	It returns the ServiceProvider element.	Empty	<ows:ServiceProvider> element instance
	<RESET Root Path>/Contents	It returns the Contents element.	Empty	<sps:Contents> element instance
	<RESET Root Path>/OperationsMetadata	It returns the OperationsMetadata element.	Empty	<ows: OperationsMetadata> element instance
	<RESET Root Path>/Notifications	It returns the Notifications element.	Empty	<sps: Notifications> element instance
POST	Not used by this document.			
PUT	Not used by this document.			
DELETE	Not used by this document.			

7.2.4 Error handling

In the event that a **RESET** server encounters an error during the accessing of the Capabilities resource, it returns an HTTP response including:

- HTTP Status Code: 4XX for errors on the client side; 5XX for errors on server side.
- HTTP Entity Body: ows:ExceptionReport element set as specified in §8 of [NR15].

The following table reports the possible error conditions with the defined HTTP responses.

Table 7-3 Capabilities Error Handling

Error Description	HTTP Error Code	OGC Exception Report		
		"exceptionCode"	**"locator"**	**"ExceptionText"**
Bad Input Parameter. E.g. an incorrect Capabilities sub-section has been specified.	400	InvalidParameterValue	Name of parameter with invalid value.	"Invalid value for Parameter"
Internal Server Error An error occurred inside the server while processing the request.	500	"NoApplicableCode"		

7.3 Procedures resource

The procedures resource contains descriptions of the sensors handled on this server, as well as the description of the associated tasking requests. It shall contain all the sensors described in the Contents section of the Capabilities resource see 7.2 above.

These description resources are split into three parts:

- Sensor description: an XML description of the sensor, including its modes, covered area, and sensor-specific parameters, in a given format, e.g. **SensorML**, etc.
- Tasking requests description: the parameters structure to be sent to task a specific sensor.
- Availability: an optional resource that provides the periods of availability of a sensor.

7.3.1 URL

Table 7-4 Procedures URL

URL	Definition
<ProcedureResourceURL>	<ResetRootPath>/procedures/{procedure}/
<Sensor Description> in canonical XML format	<ProcedureResourceURL>/{sensorDescriptionFormat}
<Sensor and Tasking Description> in OpenSearch Description Document (OSDD)	<ProcedureResourceURL>/osdd
<Tasking Description>	<ProcedureResourceURL>/tasking
<Availability>	<ProcedureResourceURL>/availability

Table 7-5 Procedures URL parameters

Names	Definition	Data type and values
{procedure}	Identifier of the sensor, or sensor type, which description is required.	xs:string (max 255 chars) URL encoded
{sensorDescriptionFormat}	XML format in which the sensor description is required.	xs:string (max 255 chars) URL encoded

7.3.2 Definition

The procedure resource contains all the information that the client might need in order to submit feasibility and planning requests. This includes:

- Description of the sensor's capabilities in various formats.
- Description of the format of the tasking request needed to task the sensor.
- Optionally, the availability of the sensor.
- Optionally, the OpenSearch Description Document based on the **OS EOSP** specification [NR23]. This specification allows the submission of feasibility requests using the OpenSearch specification.
  This alternative format has been proposed in order to support online data access scenario's in which the result of the catalogue access includes not only the metadata records matching the query, but also pre-built URLs for direct ordering & downloading of the search results.

7.3.2.1 Sensor capabilities

The DescribeSensor operation is used to obtain a detailed sensor/procedure description encoded in a certain format, for example **SensorML**. As an **SWE** service might offer such descriptions in multiple formats, or different versions of the same format, clients generally need to indicate which format they require. The formats supported by a given sensor/procedure are defined in the Capabilities of the service.

Further description of the DescribeSensor operation is available in the SWES specification [NR23].

7.3.2.2 Tasking request description

The DescribeTasking operation is used to get the XML structure that is needed to submit feasibility or planning requests. This structure contains the basic parameters such as the Area of Interest, Time of Interest, as well as instrument modes and sensor specific acquisition and validation parameters. The full description of these parameters is available in the **EO-SPS** specification [NR22].

7.3.2.3 Sensor availabilities

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

7.3.3 HTTP methods

Table 7-6 Procedures HTTP methods

HTTP Method	URL	Definition	Request Entity Body	Response Entity Body
GET	<ResetRootPath>/procedures/{procedure}/{sensorDescriptionFormat}	It returns the description of the sensor in the required format.	Empty	<eosps: eoTaskingParameters > element instance
	<ResetRootPath>/procedures/{procedure}/osdd	It returns the ServiceIdentification element.	Empty	<swes:ServiceIdentification> element instance
	<ResetRootPath>/procedures/{procedure}/tasking	It returns the ServiceProvider element.	Empty	<swes: procedureDescriptionFormat > element instance
	<ResetRootPath>/procedures/{procedure}/availability	It returns the Contents element.	Empty	<eosps:GetSensorAvailabilityResponse> element instance
POST	Not used by this document.			
PUT	Not used by this document.			
DELETE	Not used by this document.			

Table 7-7 Procedures HTTP methods parameters

Names	Definition	Data type and value
{procedure}	Identifier of the sensor, or sensor type, which description is required.	xs:string (max 255 chars) URL encoded Allowed values are described in the Capabilities.
{sensorDescriptionFormat}	XML format in which the sensor description is required.	xs:string (max 255 chars) URL encoded Allowed values are described in the Capabilities.

7.3.4 Error handling

In the event that a **RESET** server encounters an error during the accessing of the procedure description resources, it returns an HTTP response including:

- HTTP Status Code: 4XX for errors on the client side; 5XX for errors on server side.
- HTTP Entity Body: ows:ExceptionReport element set as specified in §8 of [NR15].

The following table reports the possible error conditions with the defined HTTP responses.

Table 7-8 Procedures Error handling

Error Description	HTTP Error Code	OGC Exception Report		
		"exceptionCode"	**"locator"**	**"ExceptionText"**
Missing Parameter. E.g. no sensor or sensor type has been specified.	400	MissingParameterValue	Name of missing parameter.	"Missing value for Parameter"
Bad Input Parameter. E.g. an incorrect sensor or sensor type has been specified.	400	InvalidParameterValue	Name of parameter with invalid value.	"Invalid value for Parameter"
Internal Server Error An error occurred inside the server while processing the request.	500	"NoApplicableCode"		
Authentication Fail	401	AuthenticationFailed	"identity_token"	"Invalid or missing identity information"
Authorization Fail	403	AuthorizationFailed	"procedure"	Text describing the item not authorized.

7.4 Feasibility resource

The Feasibility resource is the envelope sent from the client to calculate the feasibility of a tasking request with the selected tasking parameters. Additionally, this resource provides a means to get the status of the request and the results it produces. These results consist of segments that can then be submitted by the user as future acquisitions.

This resource contains the following elements:

- FeasibilityResourceURL(base URL): the base URI of the feasibility resource. It is also the URL where the feasibility analysis request shall be sent.
- FeasibilityTaskResourceURL: provides access to the result of a given task. It is also the URL where the user shall send the requests in order to submit some segments that are part of this task's results.

7.4.1 URL

Table 7-9 Feasibility URL

URL	Definition
<FeasibilityResourceURL> / <AllTasks>	<ResetRootPath>/feasibility/
<Task Status By ID> in canonical XML format	<FeasibilityResourceURL>/{taskId}
<TaskSegmentsByID> in canonical XML format	<FeasibilityResourceURL>/{taskId}/segments

Table 7-10 Feasibility URL parameters

Names	Definition	Data type and values
{taskId}	Identifier of the task required by the user.	xs:string (max 255 chars) URL encoded

7.4.2 Definition

The Feasibility resource implements the following operations of the **EO-SPS** specification:

- **GetFeasibility** (***POST /{ FeasibilityResourceURL}***):
  This operation starts a feasibility analysis, which is a preliminary calculation of the capabilities of a given sensor, or set of sensors, to acquire a given Area of Interest during a certain Time of Interest. This calculation can be done synchronously or asynchronously. In the case of an asynchronous process, the results of the process shall be available through a GetTask operation;
- **Get Status** (***GET /{ FeasibilityResourceURL}/{taskId}/***):
  This operation should only be implemented in the case of an asynchronous implementation. In which case, the response will provide the status of the task.
- **GetTask** (***GET /{ FeasibilityResourceURL}/{taskId}/segments/***):
  This operation retrieves the results of a feasibility analysis process.
- **SubmitSegmentByID** (***POST /{ FeasibilityResourceURL}/{taskId}/segments/***):
  This operation allows the user to submit some individual segments from the task results for a planning request.

7.4.2.1 Submitting a feasibility request

The GetFeasibility request is a **POST** request that contains the following elements:

- Synchronous: a Boolean that allows the selection of synchronous and asynchronous modes for the calculation of the feasibility analysis.
- Procedure: the sensor or sensor type that will be used to complete the request.
- Tasking parameters: the XML structure containing the parameters of the request. This includes Area of Interest, Time of Interest as well as Acquisition and Validation parameters. This structure is provided by the DescribeTasking request and is described in the **EO-SPS** specification [NR22].

The response to this request depends on the mode chosen in the request:

- In the asynchronous mode, the response will only contain a status report stating if the request has been accepted by the server or not. The response is in XML format and is based on the StatusReport defined in the SPS specification [NR22]. The status of the task will be GETFEASIBILITY_PENDING.
- In the synchronous mode, the response will also contain, in addition to the StatusReport, a FeasibilityStudy structure. This XML structure is part of the **EO-SPS** specification [NR22] and it contains a full description of a tasking request. The main part of the response is a list of Segment objects, which describe an acquisition that can be performed by a certain sensor. These segments can then be used in the SubmitSegmentByID operation to be submitted for planning. The status will be GETFEASIBILITY_COMPLETED.

7.4.2.2 Checking the status of a task

The GetStatus operation is optional and should only be implemented if the server implements asynchronous processing. In this case, it allows a user to check the status of a task. The response is based on the StatusReport of the **SPS** specification [NR22].

7.4.2.3 Getting the segments of a task

The GetTask operation provides access to the results of a feasibility task that has been completed. The response to this request is based on the same structures as the response to a GetFeasibility request:

- StatusReport: a summary of the status of the task. It contains the ID of the task, the sensor used, the status of the task and the parameters provided in the tasking request.
- FeasibilityStudy: the result of a feasibility analysis. It is composed of a list of segments. The Segment structure contains the full description of an acquisition, including the footprint describing the area covered, the time of acquisition, the sensor used and its mode. It can also be extended to contain additional information, e.g. Downlink information, cloud coverage, etc.

7.4.2.4 Submitting segments for planning

The SebmitSegmentByID operation allows the user to use the results of a feasibility analysis in order to submit the segments for planning. This request requires the following information:

- taskId: the ID of the task containing the segments that the user wants to submit.
- segmentId (multiple): a list of segment identifiers. This identifier is part of the Segment structure as defined in the **EO-SPS** specification [NR22].

This operation will create a planning task that will be available in the Planning resource 7.5 below.

The response to the operation will include the following elements:

- StatusReport: a summary of the status of the task. It contains the ID of the task, the sensor used, the status of the task and the parameters provided in the tasking request.
- ProgrammingStatus: this structure describes the results of the submission. It is similar to the FeasibilityStudy as it also composed of a list of Segments. Its full description is available in the **EO-SPS** specification [NR22].

7.4.3 HTTP methods

Table 7-11 Feasibility HTTP methods

HTTP Method	URL	Definition	Request Entity Body	Response Entity Body
GET	<ResetRootPath> /feasibility/{taskId}	Equivalent to GetStatus It returns the status of a task.	Empty	< eosps:StatusReport > element instance
	<ResetRootPath> /feasibility/{taskId}/segments	Equivalent to GetTask. It returns the results of a task.	Empty	< eosps:StatusReport > element instance < eosps:FeasibilityStudy> element instance
POST	<ResetRootPath> /feasibility?synchronous={isSynchronous}	Equivalent to GetFeasibility Permits to perform a Feasibility analysis.	EO-SPS: Tasking Parameters	< eosps:StatusReport > element instance < eosps:FeasibilityStudy> element instance if synchronous processing
	<ResetRootPath>/ feasibility/{taskId}/ segments/{segmentID*}	Equivalent to SubmitSegmentByID. It permits to submit the segments resulting from a Feasibility Analysis.	Empty	< eosps:StatusReport > element instance <eosps:FeasibilityStudy > element instance
PUT	Not used by this document.			
DELETE	Not used by this document.			

*: the user can submit more than one segment at a time

Table 7-12 Feasibility HTTP methods parameters

Names	Definition	Data type and value
{taskId}	Identifier of the sensor, or sensor type, which description is required.	xs:string (max 255 chars) URL encoded Allowed values are described in the GetTask operation.
{segmentId}	Identifier of a segment, within the given task, to be submitted for planning.	xs:string (max 255 chars) URL encoded Allowed values are described in the GetTask operation
{ isSynchronous }	A Boolean permitting the client to choose between synchronous and asynchronous processing when starting a feasibility analysis.	xs:boolean (true or false)

7.4.4 Error handling

In the event that a **RESET** server encounters an error during the accessing of the procedure description resources, it returns an HTTP response including:

- HTTP Status Code: 4XX for errors on the client side; 5XX for errors on server side.
- HTTP Entity Body: ows:ExceptionReport element set as specified in §8 of [NR15].

The following table reports the possible error conditions with the defined HTTP responses.

Table 7-13 Feasibility Error handling

Error Description	HTTP Error Code	OGC Exception Report		
		"exceptionCode"	**"locator"**	**"ExceptionText"**
Missing Parameter. E.g. no sensor or sensor type has been specified.	400	MissingParameterValue	Name of missing parameter.	"Missing value for Parameter"
Bad Input Parameter. E.g. an incorrect sensor or sensor type has been specified.	400	InvalidParameterValue	Name of parameter with invalid value.	"Invalid value for Parameter"
Internal Server Error An error occurred inside the server while processing the request.	500	"NoApplicableCode"		
Authentication Fail	401	AuthenticationFailed	"identity_token"	"Invalid or missing identity information"
Authorization Fail	403	AuthorizationFailed	"taskId"	Text describing the item not authorized.

7.5 Planning resource

The Planning resource is the envelope sent from the client to submit a planning request to the associated mission planning system. This request can be submitted in three different ways: first by reserving and confirming it, second by directly sending the tasking parameters and finally by computing a feasibility analysis and submit some of the resulting segments. Additionally, this resource gets the status of the request and the results it produces. These results consist of segments that have been submitted for acquisition. It also allows the user to submit those segments for the actual acquisition.

This resource contains the following elements:

- PlanningResourceURL (base URL): the base URI of the planning resource. It is also the URL where the planning requests shall be sent.
- PlanningTaskResourceURL: provides access to the result of a given task. It also allows cancellation of that task.
- PlanningTaskSegmentsResourceURL: provides access to the segments that this task contains. It also provides a means to validate those segments once the results are available.

7.5.1 URL

Table 7-14 Planning URL

Definition	URL
<PlanningResourceURL> / <AllTasks>	<ResetRootPath>/planning/
<Task Status By ID> in canonical XML format	< PlanningResourceURL >/{taskId}
<TaskSegmentsBy ID> in canonical XML format	< PlanningResourceURL >/{taskId}/segments
<TaskResultsBy ID> in canonical XML format	< PlanningResourceURL >/{taskId}/results
<SegmentBy ID> in canonical XML format	< PlanningResourceURL >/{taskId}/segments/{segmentId}

Table 7-15 Planning URL parameters

Names	Definition	Data type and values
{taskId}	Identifier of the task required by the user.	xs:string (max 255 chars) URL encoded

Names	Definition	Data type and values
{segmentId}	Identifier of the segment of the task required by the user.	xs:string (max 255 chars) URL encoded

7.5.2 Definition

The Planning resource implements the following operations from the **EO-SPS** specification:

- **Submit** (***POST /{PlanningResourceURL}***):
  This operation submits a tasking request for planning without using the GetFeasibility method. That means that the user directly sends its Area and Time of Interest, along with the other tasking parameters, to the server, which will submit the request directly to the mission planning.
  This process is split in two parts:
    - Firstly, the request needs to be accepted by the mission planning system. On the server, this consists of entering this data into the mission plan. Once this is done, the acquisition is **ACCEPTED** and will be acquired.
    - Then, the system needs to wait for the satellite to get the products and send them to the ground station. The acquisition is then **ACQUIRED**.
- **GetStatus** (***GET /{PlanningResourceURL}/{taskId}/***):
  This operation gets the status of a given task. This status includes the ID and status of the task, along with the time at which it is supposed to finish, the status of the last request sent, as well as the tasking parameters used to task the sensor.
- **GetTask** (***GET /{PlanningResourceURL} /{taskId}/segments/***):
  This operation retrieves the segments that have been submitted by this task. It also contains the StatusReport equivalent to the GetStatus operation response.
- **DescribeResultAccess** (***GET /{PlanningResourceURL} /{taskId}/results/***):
  This operation retrieves the results that have been created by a planning task. These results consist of a set of URIs where products can be downloaded.
- **Cancel (*DELETE /{ PlanningResourceURL }/{taskID}*):**
  This operation cancels a task that has been launched previously. This will cancel all the segments that can still be stopped (segments that have already been acquired or rejected cannot be cancelled). The server will send back a TaskingResponse that contains a StatusReport of the Cancel request, as well as the segments descriptions with their status.

7.5.2.1 Submitting a task

The Submit request is a **POST** request that contains the following elements:

- Procedure: the sensor or sensor type that will be used to complete the request.
- Tasking parameters: the XML structure containing the parameters of the request. This includes Area of Interest, Time of Interest, and Acquisition and Validation parameters. This structure is provided by the DescribeTasking request and is described in the **EO-SPS** specification [NR22].

The response to this request contains the following elements:

- StatusReport: the status of the request and the task it created after the submission to the mission planning system. The task status will be one of PLANNING_ACCEPTED or PLANNING_FAILED.
- ProgrammingStatus: This structure is the equivalent to FeasibilityStudy for a planning request. The Segment structure contains the full description of an acquisition, including the footprint describing the area covered, the time of acquisition, the sensor used and its mode. It can also be extended to contain additional information, e.g. Downlink information, cloud coverage, etc.

7.5.2.2 Checking the status of a task

The GetStatus operation is optional and should only be implemented if the server implements asynchronous processing. In this case, it allows a user to check the status of a task. The response is based on the StatusReport of the **SPS** specification [NR22].

7.5.2.3 Getting the segments of a task

The GetTask operation gets the results of a planning task that has been submitted. The response to this request is based on the same structures as the response to a Submit request:

- StatusReport: a summary of the status of the task. It contains the ID of the task, the sensor used, the status of the task and the parameters provided in the tasking request.
- ProgrammingStatus: the result of a feasibility analysis. It is composed of a list of segments. It contains a list of Segment structures, each of which describing an acquisition that has been submitted to a mission planning for a given sensor.

7.5.2.4 Getting the results of a task

This operation provides the results of a planning task once it has been completed. This means that either all segments have been acquired or some (or all) have failed at one step of the process (rejected while submitting or failed to be acquired). It will then provide the user with the URI links where he can download the products that have been created.

If no products are available, the server shall return a DataNotAvailable structure, with the following messages, depending on the reason no segments were found:

Table 7-16 Planning: Results not available

Code	Identifier	Description
1	TaskNotFound	The task requested was not found within the server.
2	DataAccessNotFound	The task was found but has not produced any results.
3	DataAccessNotReady	The task has not produced the results **YET**; but they will be available in the future.

7.5.2.5 Cancelling a task

This operation cancels a task that has been submitted previously. It will cancel all the segments of the task that can be cancelled:

Table 7-17 Planning: Segment cancellation

Status	Cancellable	Reason
POTENTIAL	Yes	This only happens in the Planning with feasibility study 6.3.1 above scenario. It means the segment has not been accepted by the planning request.
ACCEPTED	Yes	This means the segment has been entered in the Planning system, but has not been acquired yet. Unless the satellite will acquire it before the next Ground station contact, it will be cancellable.
REJECTED	No	This means the segment has been rejected by the planning system, and won't be acquired anyway.
ACQUIRED	No	This means the segment has already been acquired, which means the products are under processing, or already available.
FAILED	No	The segment should already have been acquired, but something went wrong and it did not produce any results.

7.5.2.6 Validating a segment

Once a segment has produced a result, the user can download this product and start using it. However, if the product does not meet the user's requirements, it may be necessary for mission planning to retry the acquisition. This is where the validate operation comes to use. It allows the user to tell the mission planning system that the product produced is not fit for purpose. In this event, mission planning will submit a new segment with the same parameters in order to get a satisfying result.

7.5.3 HTTP methods

Table 7-18 Planning HTTP methods

HTTP Method	URL	Definition	Request Entity Body	Response Entity Body
GET	<ResetRootPath>/ planning/{taskId}	Equivalent to GetStatus It returns the status of a task.	Empty	<eosps:StatusReport> element instance
	<ResetRootPath>/ planning/{taskId}/segments	Equivalent to GetTask. It returns the results of a task.	Empty	<eosps:StatusReport> element instance <eosps:ProgrammingStatus> element instance
	<ResetRootPath>/ planning/{taskId}/results	Equivalent to DescribeResultAccess It returns the links to the products generated by the task.	Empty	<sps:DescribeResultAccessDocument> - <sps:DataAvailable> - <sps:DataNotAvailable>
POST	<ResetRootPath>/ planning/	Equivalent to Submit Permits to perform a Planning request.	EO-SPS: Tasking Parameters	<eosps:StatusReport> element instance <eosps:ProgrammingStatus> element instance
	<ResetRootPath/ planning/{taskId}/ segments/{segmentID*}	Equivalent to Validate. It permits to validate a (set of) segment(s).	Empty	< eosps:StatusReport > element instance <eosps:ProgrammingStatus> element instance
PUT	Not used by this document.			
DELETE	<ResetRootPath>/ planning/{taskId}	Equivalent to Cancel. It permits to cancel a task that has been submitted earlier.	Empty	<eosps:StatusReport> element instance <eosps:ProgrammingStatus> element instance

*: the user can validate more than one segment at a time

Table 7-19 Planning HTTP methods parameters

Names	Definition	Data type and value
{taskId}	Identifier of the sensor, or sensor type, which description is required.	xs:string (max 255 chars) URL encoded Allowed values are described in the GetTask operation.
{segmentId}	Identifier of a segment, within the given task, to be submitted for planning.	xs:string (max 255 chars) URL encoded Allowed values are described in the GetTask operation.

7.5.4 Error handling

In the event that a **RESET** server encounters an error during the accessing of the procedure description resources, it returns an HTTP response including:

- HTTP Status Code: 4XX for errors on the client side; 5XX for errors on server side.
- HTTP Entity Body: ows:ExceptionReport element set as specified in §8 of [NR15].

The following table reports the possible error conditions with the defined HTTP responses.

Table 7-20 Planning Error handling

Error Description	HTTP Error Code	OGC Exception Report		
		"exceptionCode"	**"locator"**	**"ExceptionText"**
Missing Parameter. E.g. no sensor or sensor type has been specified.	400	MissingParameterValue	Name of missing parameter.	"Missing value for Parameter"
Bad Input Parameter. E.g. an incorrect sensor or sensor type has been specified.	400	InvalidParameterValue	Name of parameter with invalid value.	"Invalid value for Parameter"
Internal Server Error An error occurred inside the server while processing the request.	500	"NoApplicableCode"		
Authentication Fail	401	AuthenticationFailed	"identity_token"	"Invalid or missing identity information"
Authorization Fail	403	AuthorizationFailed	"taskId"	Text describing the item not authorized.

7.6 Reservation resource

The Reservation resource is the resource that implements a reservation system for the submission of planning requests. This means the user is able to reserve a slot of time in the sensors' mission planning systems, without actually submitting the task. This is useful if the user wants to explore other possibilities without risking losing this one.

This resource contains the following elements:

- ReservationResourceURL (base URL): the base URI of the reservation resource. It is also the URL where the reservation request shall be sent.
- ReservationTaskResourceURL: provides access to the description of a task, i.e. the description of the segments that would be acquired (if the task is validated). It also allows modification of the parameters of the task, and "confirm" it, which means actually submitting the planning to the mission planning system.

7.6.1 URL

Table 7-21 Reservation URL

URL	Definition
<ReservationResourceURL> / <AllTasks>	<ResetRootPath>/reservation/
<Task By ID> in canonical XML format	<ReservationResourceURL>/{taskId}

Table 7-22 Reservation URL parameters

Names	Definition	Data type and values
{taskId}	Identifier of the task required by the user.	xs:string (max 255 chars) URL encoded

7.6.2 Definition

The Reservation resource implements the following operations from the **EO-SPS** specification [NR22]:

- **Reserve (*POST /{ReservationResourceURL}*):**
  This operation reserves a task in the mission planning system for a given time. The user then has until the end of that time to confirm the submission of the request to the mission planning.
- **GetTask (*GET /{ReservationResourceURL}/{taskID}*):**
  This operation retrieves the status and segments of a task that has been reserved.
- **Update (*PUT /{ ReservationResourceURL}/{taskID}*):**
  This operation allows modification of the parameters of a task that has been reserved. This might of course change the results that the task will produce.
- **Confirm (*POST /{ ReservationResourceURL}/{taskID}*):**
  This operation validates a reserved task by actually entering it in the planning system. This will create a new task(a child of the current one) in the planning resource.
- **Cancel (*DELETE /{ReservationResourceURL }/{taskID}*):**
  If the user is not satisfied with the results, or has found a better way of fulfilling their requirements, the cancel operation cancels this reservation in order to make the time slots that have been blocked, available again.

7.6.2.1 Reserving a task

The Reserve operation is a **POST** request that contains the following elements:

- Procedure: the sensor or sensor type that will be used to complete the request.
- ReservationExpiration: the time for which the user would like to reserve the task, in seconds.
- Tasking parameters: the XML structure containing the parameters of the request. This includes Area of Interest, Time of Interest, and Acquisition and Validation parameters. This structure is provided by the DescribeTasking request and is described in the **EO-SPS** specification [NR22].

The response to this request contains the following elements:

- StatusReport: the status of the request and the task it created after the submission to the mission planning system. The task status will be one of RESERVATION_ACCEPTED or RESERVATION_FAILED.
- ReservationExpiration: the time at which the reservation will actually expire. This value shall not be bigger than the one contained in the request, in other words the task shall not be reserved for longer than the user requested.

- ProgrammingStatus: This structure is the equivalent to FeasibilityStudy for a planning request. The Segment structure contains the full description of an acquisition, including the footprint describing the area covered, the time of acquisition, the sensor used and its mode. It can also be extended to contain additional information, e.g. Downlink information, cloud coverage, etc.

7.6.2.2 Updating a task's parameters

Once the user has reserved a task, it is expected that they will explore other possibilities that might fulfill their requirements. In the case that a new set of parameters are found, two courses of action are possible:

- If the new set of parameters is close enough to the one of the reservation task (e.g. if an just an acquisition mode, or an acquisition angle has been changed), the current reservation can be updated.
- If it is too different (e.g. different area or time of interest, new sensors, etc.), then the user will have to cancel the current reservation and submit/reserve another one.

The Update operation contains the following parameters:

- taskID: the task to be changed.
- taskingParameters: the new set of parameters that shall replace the current one.

This request recalculates the results that will be produced by the new set of parameters and sends back a TaskingResponse that contains a StatusReport of the Update request, as well as the segments descriptions with their status.

7.6.2.3 Confirming a task

Once the user is content with the acquisitions that have been reserved, the task can be submitted by calling the Confirm operation. This operation contains a taskID,that represents the task to be sent to the mission planning system.

This request will create a new planning resource item (/planning/{taskID}) that is linked to the current reservation task. It returns a TaskingResponse that describes this new task and contains a StatusReport of the Confirm request, as well as the segments descriptions with their status (ACCEPTED,REJECTED).

7.6.2.4 Cancelling a task

If the user is not satisfied with the results of the task and therefore does not want to submit them for planning, the reservation can be cancelled. This will free the time slots that this task was reserving from the mission planning. The operation will return the StatusReport that confirms that the task has been cancelled. Indeed, no segments will be acquired in a reservation task, so all of them are cancellable. The only case in which this operation could fail, is if the reservation has already expired, in which case the reservation was already cancelled.

7.6.3 HTTP methods

Table 7-23 Reservation HTTP methods

HTTP Method	URL	Definition	Request Entity Body	Response Entity Body
GET	<ResetRootPath>/ reservation/{taskId}	Equivalent to GetTask. It returns the results of a task.	Empty	<eosps:StatusReport> element instance <eosps:ProgrammingStatus > element instance
POST	<ResetRootPath>/ planning/	Equivalent to Reserve Permits to reserve a Planning request.	EO-SPS: Tasking Parameters	<eosps:StatusReport> element instance <eosps:ProgrammingStatus > element instance
	<ResetRootPath/ planning/{taskId}/	Equivalent to Confirm. It permits to validate a task by sending it to the mission planning system.	Empty	< eosps:StatusReport > element instance <eosps:ProgrammingStatus > element instance
PUT	<ResetRootPath>/ reservation/{taskId}	Equivalent to Update. It permits to update the tasking parameters of a task.	EO-SPS: Tasking Parameters	< eosps:StatusReport > element instance <eosps:ProgrammingStatus > element instance
DELETE	<ResetRootPath>/ planning/{taskId}	Equivalent to Cancel. It permits to cancel a task that has been reserved earlier.	Empty	<eosps:StatusReport> element instance

Table 7-24 Reservation HTTP methods parameters

Names	Definition	Data type and value
{taskId}	Identifier of the sensor, or sensor type, which description is required.	xs:string (max 255 chars) URL encoded Allowed values are described in the GetTask operation.

7.6.4 Error handling

In the event that a **RESET** server encounters an error during the accessing of the procedure description resources, it returns an HTTP response including:

- HTTP Status Code: 4XX for errors on the client side; 5XX for errors on server side.
- HTTP Entity Body: ows:ExceptionReport element set as specified in §8 of [NR15].

The following table reports the possible error conditions with the defined HTTP responses.

Error Description	HTTP Error Code	OGC Exception Report		
		"exceptionCode"	**"locator"**	**"ExceptionText"**
Missing Parameter. E.g. no sensor or sensor type has been specified.	400	MissingParameterValue	Name of missing parameter.	"Missing value for Parameter"
Bad Input Parameter. E.g. an incorrect sensor or sensor type has been specified	400	InvalidParameterValue	Name of parameter with invalid value.	"Invalid value for Parameter"
Internal Server Error An error occurred inside the server while processing the request.	500	"NoApplicableCode"		
Authentication Fail	401	AuthenticationFailed	"identity_token"	"Invalid or missing identity information"
Authorization Fail	403	AuthorizationFailed	"taskId"	Text describing the item not authorized.

Table 7-25 Reservation Error handling

8. RESET Core Requirement Class

This section reports all the requirements a **RESET** server has to comply with for claiming the conformance with respect to the Core class. The system implemented by this class provides the base operations used in all processes, such as Tasking description, Sensor description or Task description.

This section is structured by resources: for each **RESET** resource, a dedicated sub-section describes all related requirements:

- Capabilities in §8.1;
- Procedures in §8.2;
- Tasking, which regroups feasibility, planning and reservation, in §8.3.

The following requirement applies to each of these sub-sections.

Requirements Class	
http://www.opengis.net/spec/RESET/1.0/req/Core	
Target type	**RESET** Server
Dependency	**None**
Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Core/Root** All resources of a **RESET** server shall be accessible from the following root URL: http://<hostname>:<port>/<context path>/RESET/1.0.0

8.1 Capabilities

Requirements Class	
http://www.opengis.net/spec/RESET/1.0/req/Core/Capabilities	
Target type	**RESET** Server
Dependency	**None**
Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Core/Capabilities** The Capabilities resource of a RESET server shall be accessible from the RESET root path: http://<hostname>:<port>/<context path>/RESET/1.0.0 via the HTTP GET method.

8.1.1 Service Identification

Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Core/Capabilities/ServiceIdentification** The ServiceIdentification section of Capabilities resource of a **RESET** server shall be accessible from the following path: http://<hostname>:<port>/<context path>/RESET/1.0.0ServiceIdentification via the HTTP GET method.
Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Core/Capabilities/ServiceProvider** The ServiceProvider section of Capabilities resource of a **RESET** server shall be accessible from the following path: http://<hostname>:<port>/<context path>/RESET/1.0.0/ServiceProvider via the HTTP GET method.
Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Core/Capabilities/schema** The returned Capabilities resource shall consist of an XML instance document as validated by the entity Capabilities in the reset.xsd XML Schema.

8.1.2 Contents

Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Core/Capabilities/Contents** The Contents section of Capabilities resource of a **RESET** server shall be accessible from the following path: http://<hostname>:<port>/<context path>/RESET/1.0.0/Contents via the HTTP **GET** method.
Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Core/Capabilities/Contents/contents** The returned Capabilities Contents resource shall include the same elements as the one described in the **EO-SPS** specification [NR22].
Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Core/Capabilities/Contents/resource** The returned Capabilities resource shall contain all information about the resources supported by the **RESET** Server.
Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Core/Capabilities Contents//resource/profile** The returned Capabilities resource shall advertise the support of **RESET** specification including: Capabilities/ServiceIdentification/Profile= http://www.opengis.net/def/bp/reset/1.0
Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Core/Capabilities/ Contents/resource/resourceURL** The returned Capabilities resource shall advertise the supported resources setting the element: Capabilities/Contents/resourceURL reporting the URL templates for accessing to the DescribeSensor, DescribeTasking, GetSensorAvailibility resources.
Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Core/Capabilities/ Contents/resource/SupportedClass** The returned Capabilities shall advertise the supported functionalities setting the element: Capabilities/Contents/ SupportedClass reporting the supported Requirement classes: feasibility, planning, feasbiilityPlanning, reservation, cancellation.

8.1.3 Notifications

Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Core/Capabilities/Notifications** The Notifications section of Capabilities resource of a RESET server shall be accessible from the following path: http://<hostname>:<port>/<context path>/RESET/1.0.0/Notifications via the HTTP **GET** method.
Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Core/Capabilities/Notifications/contents** The returned Capabilities Notifications resource shall include the same elements as the one described in the **EO-SPS** specification [NR22].

8.1.4 OperationsMetadata

This section has been cancelled. See Definition of the Capabilities resource at 7.2 above for more detail.

Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Core/Capabilities/OperationsMetadata** The OperationsMetadata section of Capabilities resource of a RESET server shall be accessible from the following path: http://<hostname>:<port>/<context path>/RESET/1.0.0/OperationsMetadata via the HTTP **GET** method.
Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Core/Capabilities/OperationsMetadata/contents** The returned Capabilities resource shall advertise the supported operations setting the element: Capabilities/OperationsMetadata/Operation
Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Core/Capabilities/OperationsMetadata/contents/url** The returned Capabilities resource shall give the base URL of all operations setting the element: Capabilities/OperationsMetadata/Operation/DCP/HTTP/{HTTPMethodName}/@href Capabilities/ reporting the URL templates for accessing to the Feasibility, Planning and Reservation resources as described in tables 26 and 27.

The following tables display the different resources and the associated **EO-SPS** operations, for each Requirement class.

8.1.5 URL resource distribution

Whether in OperationsMetadata or in the resourceURL structures, the following resources shall be described, depending on the classes supported by the server.

8.1.5.1 Core requirement class

Operation	Resource	Method
GetCapabilities	/{root}	GET
DescribeSensor	/procedures/{procedure}	GET
DescribeTasking	/procedures/{procedure}/tasking	GET
GetSensorAvailibility	/procedures/{procedure}/availability	GET
GetStatus	/planning/{taskID}	GET
GetTask	/planning/{taskID}/segments	GET

Table 8-1 Core operations

8.1.5.2 Feasibility requirement class

Operation	Resource	Method
GetFeasibility	/feasibility	POST

Table 8-2 Feasibility operations

8.1.5.3 Planning requirement class

Operation	Resource	Method
Submit	/planning	POST
DescribeResultAccess	/planning/{taskID}/results	GET

Table 8-3 Planning operations

8.1.5.4 Feasibility Planning requirement class

Operation	Resource	Method
SubmitSegmentByID	/feasibility/{taskID}	POST

Table 8-4 Feasibility planning operations

8.1.5.5 Reservation requirement class

Operation	Resource	Method
Reserve	/reservation	POST
Update	/reservation/{taskID}	PUT
Confirm	/reservation/{taskID}	POST

Table 8-5 Reservation operations

8.1.5.6 Cancellation requirement class

Operation	Resource	Method
Cancel	/{resource*}/{taskID}	DELETE

Table 8-6 Cancellation operations

*where “resource” can be “planning” or “reservation“.

8.2 Procedures

Requirements Class	
http://www.opengis.net/spec/RESET/1.0/req/Core/Procedures	
Target type	**RESET** Server
Dependency	**None**
Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Core/Procedures** The Procedures resource of a RESET server shall be accessible from the following root path: http://<hostname>:<port>/<context path>/RESET/1.0.0/procedures via the HTTP **GET** method.

8.2.1 Tasking description

Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Core/Procedures/TaskingDescription** The DescribeTasking implementation shall be accessible from the following path: http://<hostname>:<port>/<context path>/RESET/1.0.0/procedures/{procedure}/tasking via the HTTP **GET** method.
Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Core/Procedures/TaskingDescription/contents** The DescribeTasking implementation shall describe the parameters structure defined in the **EO-SPS** specification [NR22].

8.2.2 Sensor description

Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Core/Procedures/SensorDescription** The DescribeSensor implementation shall be accessible from the following path: http://<hostname>:<port>/<context path>/RESET/1.0.0/procedures/{procedure}/tasking via the HTTP **GET** method.
Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Core/Procedures/SensorDescription/contents** The DescribeSensor implementation shall describe the sensor description structure defined in the **SWES** Specification [NR23].

8.2.3 Sensor availabilities

Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Core/Procedures/SensorAvailabilities** The GetSensorAvailabilities implementation shall be accessible from the following path: http://<hostname>:<port>/<context path>/RESET/1.0.0/procedures/{procedure}/availabilities via the HTTP **GET** method.
Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Core/Procedures/SensorAvailabilities/contents** The GetSensorAvailabilities implementation shall provide the sensor's availabilities as defined in the **EO-SPS** specification [NR22].

8.3 Tasking

8.3.1 Task status

Requirements Class	
http://www.opengis.net/spec/RESET/1.0/req/Core/Tasking	
Target type	**RESET** Server
Dependency	**None**
Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Core/Tasking/taskStatus** The GetStatus implementation shall be accessible from the following path: http://<hostname>:<port>/<context path>/RESET/1.0.0/{taskType*}/{taskID} via the HTTP **GET** method.
Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Core/Tasking/taskStatus/content** The GetStatus implementation shall provide the task's status based on the <eosps:StatusReport> defined in the **EO-SPS** specification [NR22].
Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Core/Tasking/taskStatus/content/parameters** The GetStatus implementation shall provide the tasking parameters based on the <eosps:TaskingParameters> defined in the **EO-SPS** specification [NR22]..

*where taskType is one of: *feasibility, planning, reservation.*

8.3.2 Task description

Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Core/Tasking/taskDescription** The GetTask implementation shall be accessible from the following path: http://<hostname>:<port>/<context path>/RESET/1.0.0/{taskType*}/{taskID}/segments via the HTTP GET method.
Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Core/Tasking/taskDescription/content** The GetTask implementation shall provide the same <eosps:StatusReport> as described in the Task status operation section 8.3.1 above.

*where taskType is one of: *feasibility, planning, reservation.*

9. RESET Feasibility Requirement Class

This section reports all the requirements a **RESET** server has to comply with for claiming conformance with respect to the Feasibility class. This class shall implement all functionalities that perform a feasibility analysis process.

This section is structured by resources: for each **RESET** resource, a dedicated sub-section describes all related requirements:

- Capabilities in §9.1;
- Procedures in §9.2;
- Feasibility in §9.3.

Requirements Class	
http://www.opengis.net/spec/RESET/1.0/req/Feasibility	
Target type	**RESET** Server
Dependency	**http://www.opengis.net/spec/RESET/1.0/req/Core**

9.1 Capabilities

None.

9.2 Procedures

None.

9.3 Feasibility

9.3.1 Creation

Requirements Class	
http://www.opengis.net/spec/RESET/1.0/req/Feasibility/FeasibilityAnalysis	
Target type	**RESET** Server
Dependency	**None**
Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Feasibility/FeasibilityAnalysis/creation** The RESET Server shall allow Feasibility resource creation by: - HTTP **POST** request to the Feasibility resources root URL: http://<hostname>:<port>/<context path>/RESET/1.0.0/feasibility - HTTP Entity Body including a <eosps:TaskingParameters> element as validated by eospsTaskingParameters.xsd XML Schema and provided by the Tasking description operation of the Core implementation.

Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Feasibility/FeasibilityAnalysis/creation/content**
	
	On successful FeasibilityStudy creation, a **RESET** server shall return:
	
	- HTTP status code: 200 ok;
	- HTTP entity body:
	  - <eosps:StatusReport> as described in the Task status operation section 8.3.1 above;
	  - <eosps:FeasibilityStudy> as described in the **EO-SPS** specification [NR22].

Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Feasibility/FeasibilityAnalysis/creation/content/status** On successful Feasibility Study creation, the StatusReport shall contain the "FEASIBILITY COMPLETED" status identifier.
Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Feasibility/FeasibilityAnalysis/creation/content/segments** On successful Feasibility Study, the segments provided in the response shall provide the following status: POTENTIAL, as defined in the **EO-SPS** specification [NR22].

9.3.2 Task description

Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Feasibility/FeasibilityAnalysis/taskDescription/content** The GetTask implementation shall provide the same <eosps:FeasibilityStudy> as described in the Creation operation in section 9.3.1.

10. RESET Planning Requirement Class

This section reports all the requirements a **RESET** server has to comply with for claiming conformance with respect to the Planning class. This class shall implement all functionalities that perform a planning process.

This section is structured by resources: for each **RESET** resource, a dedicated sub-section describes all related requirements:

- Capabilities in §10.1;
- Procedures in §10.2;
- Planning in §10.3.

Requirements Class	
http://www.opengis.net/spec/RESET/1.0/req/Planning	
Target type	**RESET** Server
Dependency	**http://www.opengis.net/spec/RESET/1.0/req/Core**

10.1 Capabilities

None.

10.2 Procedures

None.

10.3 Planning

10.3.1 Status

The task and segment status during a planning request are defined hereby:

Status Identifier	Description	Is Final State
PLANNING ACCEPTED	The planning request has been accepted by the server.	No
PLANNING FAILED	The planning request has been rejected by the server, or failed to produce results.	Yes
PLANNING COMPLETED	The planning has finished.	Yes
PLANNING CANCELLED	The planning has been cancelled by the user (see Task cancellation in section 13.3.1).	Yes

Table 10-1 Planning task status description

Status Identifier	Description	Is Final State
ACCEPTED	The segment has been submitted.	No
REJECTED	The segment has been rejected on submission.	Yes
ACQUIRED	The segment has been acquired.	Yes
FAILED	The segment has failed to be acquired.	Yes
CANCELLED	The segment acquisition has been cancelled by the user.	Yes

Table 10-2 Planning segments status description

10.3.2 Creation

Requirements Class	
http://www.opengis.net/spec/RESET/1.0/req/Planning/Planning	
Target type	**RESET** Server
Dependency	**None**
Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Planning/Planning/creation** The RESET Server shall allow Feasibility resource creation by: - HTTP **POST** request to the Feasibility resources root URL: http://<hostname>:<port>/<context path>/RESET/1.0.0/feasibility - HTTP Entity Body including a <eosps:TaskingParameters> element as validated by eospsTaskingParameters.xsd XML Schema and provided by the Tasking description operation of the Core implementation in section 8.2.1.

Requirement	**http://www.opengis.net/spec/RESET/1.0/req/ Planning/Planning/creation/content**
	
	On successful Planning creation, a **RESET** server shall return:
	
	- HTTP status code: 200 ok;
	- HTTP entity body:
	  - <eosps:StatusReport> as described in the Task status operation section 8.3.1 above;
	  - <eosps:ProgrammingStatus> as described in the **EO-SPS** specification [NR22].

Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Planning/Planning/creation/content/status** On successful planning creation, the StatusReport shall contain the "PLANNING ACCEPTED" status identifier. Otherwise, "PLANNING FAILED" shall be used.
Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Planning/Planning/creation/content/segments** On successful Planning submission, the status of the segments provided in the ProgrammingStatus shall be one of: - "ACCEPTED": the segment has been planned - "REJECTED": the segment could not be planned

10.3.3 Acquisition

This is not an action triggered by the client/user but by the mission planning system. It happens when one or more segments have been acquired by the satellite and downloaded into a ground station. It means the end of the process for this segment, unless the user has requested some extra processing.

Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Planning/Planning/acquisition/content/** When the planning is finished, the task status shall change to "PLANNING COMPLETED". However, if no segments have been acquired, the task is considered as failed, therefore its status shall be "PLANNING FAILED".
Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Planning/Planning/acquisition/content/segments/** On successful acquisition, the segment status shall change to "ACQUIRED". If the acquisition fails, the status shall change to "FAILED".

10.3.4 Task description

Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Planning/Planning/taskDescription/content** The GetTask implementation shall provide the same <eosps:StatusReport> and <eosps: ProgrammingStatus > as described in the Creation operation section 0.

10.3.5 Result description

Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Planning/Planning/taskResultDescription/** The DescribeResultAccess implementation shall be accessible from the following path: http://<hostname>:<port>/<context path>/RESET/1.0.0/{taskType*}/{taskID}/results via the HTTP **GET** method.
Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Planning/Planning/taskResultDescription/content** The DescribeResultAccess implementation shall provide the same <sps:DescribeResultAccessResponse> as described in the **SPS** specification [NR22].

10.3.6 Segments validation

Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Planning/Planning/submission** The **RESET** Server shall allow validation of a Planning resource by: - HTTP **POST** request to the Planning resource URL: http://<hostname>:<port>/<context path>/RESET/1.0.0/planning/{taskID} where taskID is a path parameter - HTTP query parameters including several {segmentID} elements.

Requirement	**http://www.opengis.net/spec/RESET/1.0/req/FeasibilityPlanning/FeasibilityAnalysis/submission/content**
	
	On successful Planning validation, a **RESET** server shall return:
	
	- HTTP status code: 200 OK;
	- HTTP entity body:
	  - <eosps:StatusReport> as described in the Task status operation section 8.3.1 above;
	  - <eosps:ProgrammingStatus> as described in the Creation section 0 of the RESET Planning Requirement Class section 10.

Requirement	**http://www.opengis.net/spec/RESET/1.0/req/FeasibilityPlanning/FeasibilityAnalysis/submission/content/status** On successful Planning validation, and only if all segments are in final state, the StatusReport shall contain the "PLANNING VALIDATED" status identifier.
Requirement	**http://www.opengis.net/spec/RESET/1.0/req/FeasibilityPlanning/FeasibilityAnalysis/submission/content/segments** On successful Planning validation, the status of the segments provided in the ProgrammingStatus shall be: - "VALIDATED": the segment has been validated.

11. RESET Feasibility Planning Requirement Class

This section reports all the requirements a **RESET** server has to comply with for claiming conformance with respect to the Feasibility Planning class. This class shall implement all functionalities that perform a feasibility analysis process and then submit the results of this process.

This section is structured by resources: for each **RESET** resource, a dedicated sub-section describes all related requirements:

- Capabilities in §11.1;
- Procedures in §11.211.2 ;
- Feasibility in §11.3;
- Planning in §11.4.

Requirements Class	
http://www.opengis.net/spec/RESET/1.0/req/FeasibilityPlanning	
Target type	**RESET** Server
Dependency	**http://www.opengis.net/spec/RESET/1.0/req/Core** **http://www.opengis.net/spec/RESET/1.0/req/Feasibility** **http://www.opengis.net/spec/RESET/1.0/req/Planning**

11.1 Capabilities

None.

11.2 Procedures

None.

11.3 Feasibility

11.3.1 Submission

Requirement	**http://www.opengis.net/spec/RESET/1.0/req/FeasibilityPlanning/FeasibilityAnalysis/ submission** The RESET Server shall allow submission of a Feasibility resource by: - HTTP **POST** request to the Feasibility resource URL: http://<hostname>:<port>/<context path>/RESET/1.0.0/feasibility/{taskID} where taskID is a path parameter - HTTP path/query parameters including several {segmentID} elements.

Requirement	**http://www.opengis.net/spec/RESET/1.0/req/FeasibilityPlanning/FeasibilityAnalysis/submission/content**
	
	On successful FeasibilityStudy submission, a **RESET** server shall return:
	
	- HTTP status code: 201 created;
	- HTTP entity body:
	  - <eosps:StatusReport> as described in the Task status operation section 8.3.1 above;
	  - <eosps:ProgrammingStatus> as described in the Creation section 0 of the RESET Planning Requirement Class in section 10.

Requirement	**http://www.opengis.net/spec/RESET/1.0/req/FeasibilityPlanning/FeasibilityAnalysis/submission/content/status** On successful Feasibility Study submission, the server shall use the same status identifiers as in the Creation section 0 and Acquisition section 10.3.3 of the RESET Planning Requirement Class in section 10.
Requirement	**http://www.opengis.net/spec/RESET/1.0/req/FeasibilityPlanning/FeasibilityAnalysis/submission/content/segments** On successful Planning submission, the status of the segments provided in the ProgrammingStatus shall be one of: - "ACCEPTED": the segment has been planned - "REJECTED": the segment could not be planned

11.4 Planning

None.

12. RESET Reservation Requirement Class

This section reports all the requirements a **RESET** server has to comply with for claiming conformance with respect to the Reservation class. This class shall implement all functionalities that perform a feasibility analysis process and then submit the results of this process.

This section is structured by resources: for each **RESET** resource, a dedicated sub-section describes all related requirements:

- Capabilities in §12.1;
- Procedures in §12.2;
- Planning in §12.3;
- Reservation in §12.4.

Requirements Class	
http://www.opengis.net/spec/RESET/1.0/req/Reservation	
Target type	**RESET** Server
Dependency	**http://www.opengis.net/spec/RESET/1.0/req/Core** **http://www.opengis.net/spec/RESET/1.0/req/Planning**

12.1 Capabilities

Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Reservation/Capabilities/OperationsMetadata** The OperationsMetadata of the Capabilities document shall advertise the following optional operations: *reserve, update, confirm*, as defined in OperationsMetadata section 8.1.4 of the RESET Core Requirement Class section 8.

12.2 Procedures

None.

12.3 Planning

None.

12.4 Reservation

12.4.1 Creation

Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Reservation/Reservation/creation**
	
	The **RESET** Server shall allow Reservation resource creation by:
	
	- HTTP **POST** request to the Reservation resource root URL: http://<hostname>:<port>/<context path>/RESET/1.0.0/reservation
	- HTTP Entity Body including
	  - <eosps:TaskingParameters> element as validated by eospsTaskingParameters.xsd XML Schema and provided by the Tasking description section 8.2.1 operation of the Core implementation.
	  - <sps:ReservationEspiration> as described in the **SPS** specification [NR21], in the Reserve operation section.

Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Reservation/Reservation/creation/content**
	
	On successful Reservation creation, a **RESET** server shall return:
	
	- HTTP status code: 200 ok;
	- HTTP entity body:
	  - <eosps:StatusReport> as described in the Task status operation section 8.3.1 above;
	  - <sps: reservationExpiration> as described in the **SPS** specification [NR21], in the Reserve operation section.
	  - <eosps:ProgrammingStatus> as described in the Creation section 0 of the RESET Planning Requirement Class in section 10.

Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Reservation/Reservation/creation/content/status** On successful Reservation creation, the StatusReport shall contain the "PLANNING RESERVED" status identifier.
Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Reservation/Reservation/creation/content/segments** On successful Reservation creation, the segments provided in the response shall provide the following status: POTENTIAL, as defined in the EO-SPS specification [NR22].

12.4.2 Modification

Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Reservation/Reservation/modification** The **RESET** Server shall allow Reservation resource modification by: HTTP **PUT** request to the following Reservation resource URL: http://<hostname>:<port>/<context path>/RESET/1.0.0/reservation HTTP Entity Body including o <eosps:TaskingParameters> element as validated by eospsTaskingParameters.xsd XML Schema and provided by the Tasking description operation section 8.2.1 of the Core implementation. o <sps:ReservationEspiration> as described in the **SPS** specification [NR21], in the Reserve operation section.
Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Reservation/Reservation/modification/content** On successful Reservation modification, a **RESET** server shall return: - HTTP status code: 202 accepted; - HTTP entity body: o <eosps:StatusReport> as described in the Task status operation section 8.3.1 above; o <sps: reservationExpiration> as described in the SPS specification [NR21], in the Reserve operation section. o <eosps:ProgrammingStatus> as described in the Creation section 0 of the RESET Planning Requirement Class section 10.

12.4.3 Submission

Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Reservation/Reservation/submission** The **RESET** Server shall allow Reservation resource submission by: HTTP **PUT** request to the following Reservation resource URL: http://<hostname>:<port>/<context path>/RESET/1.0.0/reservation HTTP Entity Body including o <eosps:TaskingParameters> element as validated by eospsTaskingParameters.xsd XML Schema and provided by the Tasking description operation section 8.2.1 of the Core implementation.
Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Reservation/Reservation/submission/content** On successful Reservation submission, a **RESET** server shall return: - HTTP status code: 200 ok; - HTTP entity body: o <eosps:StatusReport> as described in the Task status operation section 8.3.1 above; o <eosps:ProgrammingStatus> as described in the Creation section 0 of the RESET Planning Requirement Class in section 10.

Requirement	http://www.opengis.net/spec/RESET/1.0/req/FeasibilityPlanning/FeasibilityAnalysis/ submission/content/status On successful Reservation submission, the server shall use the same status identifiers as in the Creation section 0 and Acquisition section 10.3.3 of the RESET Planning Requirement Class section 10.

13. RESET Cancellation Requirement Class

This section reports all the requirements a **RESET** server has to comply with for claiming conformance with respect to the Cancellation class. This class shall implement all functionalities that cancel a planning or reservation process.

This section is structured by resources: for each **RESET** resource, a dedicated sub-section describes all related requirements:

- Capabilities in §13.1;
- Procedures in §13.2;
- Tasking in §13.3.

Requirements Class	
http://www.opengis.net/spec/RESET/1.0/req/Cancellation	
Target type	**RESET** Server
Dependency	**http://www.opengis.net/spec/RESET/1.0/req/Core** **http://www.opengis.net/spec/RESET/1.0/req/Planning**

13.1 Capabilities

Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Cancellation/Capabilities/OperationsMetadata** The OperationsMetadata of the Capabilities document shall advertise the following optional operation: *cancel*, as defined in OperationsMetadata section 8.1.4 of the RESET Core Requirement Class in section 8.

13.2 Procedures

None.

13.3 Tasking

13.3.1 Task cancellation

Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Cancellation/TaskCancellation** The **RESET** Server shall allow cancellation of a Planning or Reservation resource by: - HTTP **DELETE** request to the Planning or Reservation resource root URL: http://<hostname>:<port>/<context path>/RESET/1.0.0/{taskType*}/{taskID} where taskID is a path parameter - HTTP query parameters including several {segmentID} elements.

Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Cancellation/TaskCancellation/content**
	
	On successful Planning or Reservation cancellation, a **RESET** server shall return:
	
	- HTTP status code: 200 ok;
	- HTTP entity body:
	  o <eosps:StatusReport> as described in the Task status operation section 8.3.1 above;
	  o <eosps:ProgrammingStatus> as described in the Creation section 0 of the RESET Planning Requirement Class in section 10.

Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Cancellation/TaskCancellation/content/status** On successful Planning or Reservation cancellation, the StatusReport shall contain the "PLANNING CANCELLED" status identifier.
Requirement	**http://www.opengis.net/spec/RESET/1.0/req/Cancellation/TaskCancellation/content/status/segments** On successful Planning or Reservation cancellation, all segments cancelled shall contain the "CANCELLED" status identifier.

*where taskType as one of: *planning, reservation*

14. Annex A: Conformance Class Abstract Test Suite (Normative)

This section describes the Abstract Test Suite (ATS) of the

> **OGC RESTful** encoding of Sensor Planning Service for Earth Observation satellite Tasking.

An **ATS** provides a basis for developing an executable test suite (ETS) to verify that an Implementation Under Test (IUT) conforms to all relevant functional specifications.

A number of Conformance classes have been defined for verifying the various Requirement classes defined in this specification:

Requirement Class	Conformance Class	Conformance Class URI
Core	Core	http://www.opengis.net/spec/RESET/1.0/conf/Core
Feasibility	Feasibility	http://www.opengis.net/spec/RESET/1.0/conf/Feasibility
Planning	Planning	http://www.opengis.net/spec/RESET/1.0/conf/Planning
Reservation	Reservation	http://www.opengis.net/spec/RESET/1.0/conf/Reservation
Cancellation	Cancellation	http://www.opengis.net/spec/RESET/1.0/conf/Cancellation

Table 14-1 Conformance classes

Each Conformance Class is composed of a set of tests, each verifying one or more requirements of the corresponding Requirements Class.

Each Conformance Class covers all requirements of the corresponding Requirements Class.

It has to be noted that the tests reported in the Conformance classes have “temporal dependencies”: in fact for running one test another specific test might be needed (e.g. to test tasking status at least one task needs to be submit in the **RESET** Server). Then:

- The tests specified in a Conformance Class must be executed in the order they are specified in the document;
- The tests of a Conformance Class can be started only if the tests of the parent class have been completed.

14.1 Conformance class: Core

14.1.1 Capabilities resource

14.1.1.1 GET Capabilities

a) Test id: http://www.opengis.net/spec/RESET/1.0/conf/Core/GET_Capabilities

b) Test purpose: To verify that the **RESET** Server under test correctly supports the HTTP **GET** method on Capabilities resource.

c) Test method:

Verify that the **RESET** Server under test accepts the following request:

- HTTP Method: GET
- PATH: http://<hostname>:<port>/<context path>/RESET/1.0.0

Verify the response message:

- HTTP Status: 200 OK
- HTTP Entity Body:
  - complies with the Capabilities element of reset.xsd schema
  - the ows:OperationsMetadata element is filled-in with the list of supported operations
  - the Capabilities/ServiceIdentification/Profile element is set with: http://www.opengis.net/def/bp/RESET/1.0
  - The Capabilities/Contents shall report the supported resources
  - all attributes of Capabilities/Contents/GetStatusCapabilities element are set to true
  - at least one sensor identifier is returned

Pass if the assertion is satisfied; fail otherwise.

d) References:

- http://www.opengis.net/spec/RESET/1.0/req/Core/Root
- http://www.opengis.net/spec/RESET/1.0/req/Core/Capabilities
- http://www.opengis.net/spec/RESET/1.0/req/Core/Capabilities/schema
- http://www.opengis.net/spec/RESET/1.0/req/Core/Capabilities/resource
- http://www.opengis.net/spec/RESET/1.0/req/Core/Capabilities/resource/profile
- http://www.opengis.net/spec/RESET/1.0/req/Core/Capabilities/Contents
- http://www.opengis.net/spec/RESET/1.0/req/Core/Capabilities/Contents/contents
- http://www.opengis.net/spec/RESET/1.0/req/Core/Capabilities/Notifications
- http://www.opengis.net/spec/RESET/1.0/req/Core/Capabilities/Notifications/contents
- http://www.opengis.net/spec/RESET/1.0/req/Core/Capabilities/OperationsMetdata
- http://www.opengis.net/spec/RESET/1.0/req/Core/Capabilities/OperationsMetdata/contents

- http://www.opengis.net/spec/RESET/1.0/req/Core/Capabilities/OperationsMetadata/contents/url

e) **Test type**: Capability

14.1.1.2 GET Capabilities/ServiceIdentification

a) **Test id:**
http://www.opengis.net/spec/RESET/1.0/conf/Core/GET_Capabilities/ServiceIdentification

b) **Test purpose**: To verify that the **RESET** Server under test correctly returns the ServiceIdentification section of a Capabilities resource.

c) **Test method**:

Verify that the **RESET** Server under test accepts the following request:

- HTTP Method: GET
- PATH:
  http://<hostname>:<port>/<context path>/RESET/1.0.0/ServiceIdentification

Verify the response message:

- HTTP Status: 200 OK
- HTTP Entity Body:
  - complies with the ServiceIdentification element of ows.xsd schema

Pass if the assertion is satisfied; fail otherwise.

d) **References**:

- http://www.opengis.net/spec/RESET/1.0/req/Core/Capabilities/ServiceIdentification

e) **Test type**: Capability

14.1.1.3 GET Capabilities/ServiceProvider

a) **Test id:**

http://www.opengis.net/spec/RESET/1.0/conf/Core/GET_Capabilities/ServiceProvider

b) **Test purpose**: To verify that the **RESET** Server under test correctly returns the ServiceProvider section of a Capabilities resource.

c) **Test method**:

Verify that the **RESET** Server under test accepts the following request:

- HTTP Method: GET
- PATH: http://<hostname>:<port>/<context path>/RESET/1.0.0/ServiceProvider

Verify the response message:

- HTTP Status: 200 OK
- HTTP Entity Body:

- complies with the ServiceProvider element of ows.xsd schema

Pass if the assertion is satisfied; fail otherwise.

d) References:

- http://www.opengis.net/spec/RESET/1.0/req/Core/Capabilities/ServiceProvider

e) Test type: Capability

14.1.1.4 GET Capabilities/Contents

a) Test id:

http://www.opengis.net/spec/RESET/1.0/conf/Core/GET_Capabilities/Contents

b) Test purpose: To verify that the **RESET** Server under test correctly returns the Contents section of a Capabilities resource.

c) Test method:

Verify that the **RESET** Server under test accepts the following request:

- HTTP Method: GET
- PATH: http://<hostname>:<port>/<context path>/RESET/1.0.0/Contents

Verify the response message:

- HTTP Status: 200 OK
- HTTP Entity Body:
  - complies with the Contents element of RESET.xsd schema
  - at least one sensor identifier is returned.

Pass if the assertion is satisfied; fail otherwise.

d) References:

- http://www.opengis.net/spec/RESET/1.0/req/Core/Capabilities/Contents

e) Test type: Capability

14.1.1.5 GET Capabilities/OperationsMetadata

a) Test id:

http://www.opengis.net/spec/RESET/1.0/conf/Core/GET_Capabilities/OperationsMetadata

b) Test purpose: To verify that the **RESET** Server under test correctly returns the OperationsMetadata section of a Capabilities resource.

c) Test method:

Verify that the **RESET** Server under test accepts the following request:

- HTTP Method: GET
- PATH: http://<hostname>:<port>/<context path>/RESET/1.0.0/OperationsMetadata

Verify the response message:

- HTTP Status: 200 OK
- HTTP Entity Body:
  - complies with the Contents element of reset.xsd schema
  - at least the operations of the Core, plus one of Feasibility or Planning, classes is implemented.

Pass if the assertion is satisfied; fail otherwise.

d) References:

- http://www.opengis.net/spec/RESET/1.0/req/Core/Capabilities/Contents

e) Test type: Capability

14.1.2 Procedures resource

14.1.2.1 GET Sensor description

a) Test id:

http://www.opengis.net/spec/RESET/1.0/conf/Core/GET_procedures/{procedure}/{sensorDescriptionFormat}

b) Test purpose: To verify that the **RESET** Server under test correctly returns the description of the Sensor in the sensorDescriptionFormat asked in the request.

c) Test method:

Verify that the **RESET** Server under test accepts the following request:

- HTTP Method: GET
- PATH:
  http://<hostname>:<port>/<context path>/RESET/1.0.0/procedures/{procedure}/{sensorDescriptionFormat}

Verify the response message:

- HTTP Status: 200 OK
- HTTP Entity Body:
  - complies with the sensor description format asked in the request.

Pass if the assertion is satisfied; fail otherwise.

d) References:

- **http://www.opengis.net/spec/RESET/1.0/req/Core/Procedures/SensorDescription/**
- **http://www.opengis.net/spec/RESET/1.0/req/Core/Procedures/SensorDescription/contents**

e) Test type: Capability

14.1.2.2 GET Tasking request description

a) Test id:

http://www.opengis.net/spec/RESET/1.0/conf/Core/GET_procedures/{procedure}/tasking

b) Test purpose: To verify that the **RESET** Server under test correctly returns the description of the tasking request for the procedure(sensor or sensor type) asked in the request.

c) **Test method**:

Verify that the **RESET** Server under test accepts the following request:

- HTTP Method: GET
- PATH:
  http://<hostname>:<port>/<context path>/RESET/1.0.0/procedures/{procedure}/tasking

Verify the response message:

- HTTP Status: 200 OK
- HTTP Entity Body:
  - complies with the tasking parameters described in the **EO-SPS** specification [NR22];
  - choices given for the parameters correspond to the procedure's capabilities.

Pass if the assertion is satisfied; fail otherwise.

d) **References**:

- **http://www.opengis.net/spec/RESET/1.0/req/Core/Procedures/TaskingDescription/**
- **http://www.opengis.net/spec/RESET/1.0/req/Core/Procedures/TaskingDescription/contents**

e) **Test type**: Capability

14.1.2.3 GET Sensor Availibilities

a) **Test id:**

http://www.opengis.net/spec/RESET/1.0/conf/Core/GET_procedures/{procedure}/availibilities

b) **Test purpose**: To verify that the **RESET** Server under test correctly returns the description of the tasking request for the procedure (sensor or sensor type) asked in the request.

c) **Test method**:

Verify that the **RESET** Server under test accepts the following request:

- HTTP Method: GET
- PATH:
  http://<hostname>:<port>/<context path>/RESET/1.0.0/procedures/{procedure}/availibilities

Verify the response message:

- HTTP Status: 200 OK
- HTTP Entity Body:
  - complies with the GetSensorAvailibilityResponse described in the **EO-SPS** specification [NR22].

Pass if the assertion is satisfied; fail otherwise.

d) **References**:

- **http://www.opengis.net/spec/RESET/1.0/req/Core/Procedures/ SensorAvailabilities /**
- **http://www.opengis.net/spec/RESET/1.0/req/Core/Procedures/ SensorAvailabilities /contents**

e) Test type: Capability

14.1.3 Tasking resources

None.

14.2 Conformance class: Feasibility

14.2.1 Feasibility resource

14.2.1.1 POST Feasibility

a) Test id:

http://www.opengis.net/spec/RESET/1.0/conf/ Feasibility /POST_Feasibility/

b) Test purpose: To verify whether the **RESET** Server under test is able to manage HTTP **POST** requests on Feasibility resource (i.e. it is able to create Feasibility resources).

Note that the request message shall be prepared in order to allow the RESET Server under test to accept the task (i.e. it has to be created using GET Tasking request description response in section 14.5.2.2).

c) Test method:

Verify that the **RESET** Server under test accepts the following request:

- HTTP Method: POST
- PATH: http://<hostname>:<port>/<context path>/RESET/1.0.0/feasibility
- ENTITY: <eosps:TaskingParameters>

Verify the response message:

- HTTP Status: 200 OK
- HTTP Entity Body:
  - complies with the GetFeasibilityResponse described in the **EO-SPS** specification [NR22];
  - contains a StatusReport element as defined in RESET Feasibility Requirement Class section 9 above;
  - StatusReport contains a "FEASIBILITY COMPLETED" status identifier;
  - contains at least one FeasibilityStudy element as defined in RESET Feasibility Requirement Class section 9 above;
  - Segment structures within the FeasibilityStudy have a "POTENTIAL" status.

Pass if the assertion is satisfied; fail otherwise.

d) References:

- **http://www.opengis.net/spec/RESET/1.0/req/Feasibility/FeasibilityAnalysis/ creation**
- **http://www.opengis.net/spec/RESET/1.0/req/Feasibility/FeasibilityAnalysis/ creation/content**
- **http://www.opengis.net/spec/RESET/1.0/req/Feasibility/FeasibilityAnalysis/ creation/content/status**
- **http://www.opengis.net/spec/RESET/1.0/req/Feasibility/FeasibilityAnalysis/ creation/content/segments**

e) Test type: Capability

14.2.1.2 GET TaskStatus

a) Test id:

http://www.opengis.net/spec/RESET/1.0/conf/ Feasibility /GET_Feasibility/taskID/

b) Test purpose: To verify whether the **RESET** Server under test is able to retrieve the status of a previously created Feasibility resource.

Note that the test must be prepared inserting an existing task id (using the POST Feasibility operation section 14.4.1.1) and not using an arbitrary value.

c) Test method:

Verify that the **RESET** Server under test accepts the following request:

- HTTP Method: GET
- PATH:
  http://<hostname>:<port>/<context path>/RESET/1.0.0/feasibility/{taskID}

Verify the response message:

- HTTP Status: 200 OK
- HTTP Entity Body:
  - complies with the GetStatusResponse described in the **EO-SPS** specification [NR22];
  - contains a StatusReport element as defined in RESET Feasibility Requirement Class section 9 above;
  - StatusReport contains a "FEASIBILITY COMPLETED" status identifier.

Pass if the assertion is satisfied; fail otherwise.

d) References:

- **http://www.opengis.net/spec/RESET/1.0/req/Core/Tasking/taskStatus**
- **http://www.opengis.net/spec/RESET/1.0/req/Core/Tasking/taskStatus/content**
- **http://www.opengis.net/spec/RESET/1.0/req/Core/Tasking/taskStatus/content/parameters**
- **http://www.opengis.net/spec/RESET/1.0/req/Feasibility/FeasibilityAnalysis/creation/content/status**

e) Test type: Capability

14.2.1.3 GET TaskDescription

a) Test id:

http://www.opengis.net/spec/RESET/1.0/conf/Feasibility/GET_Feasibility/taskID/segments/

b) Test purpose: To verify whether the **RESET** Server under test is able to retrieve the status and segments of a previously created Feasibility resource.

Note that the test must be prepared inserting an existing task id (using the POST Feasibility operation section 14.4.1.1) and not using an arbitrary value.

c) Test method:

Verify that the **RESET** Server under test accepts the following request:

- HTTP Method: GET

- PATH:
  http://<hostname>:<port>/<context
  path>/RESET/1.0.0/feasibility/{taskID}/segments

Verify the response message:

- HTTP Status: 200 OK
- HTTP Entity Body:
  - complies with the GetTaskResponse described in the **EO-SPS** specification [NR22];
  - contains a StatusReport element as defined in RESET Feasibility Requirement Class section 9 above;
  - StatusReport contains a "FEASIBILITY COMPLETED" status identifier;
  - contains at least one FeasibilityStudy element as defined in RESET Feasibility Requirement Class section 9 above;
  - Segment structures within the FeasibilityStudy have a "POTENTIAL" status.

Pass if the assertion is satisfied; fail otherwise.

d) References:

- **http://www.opengis.net/spec/RESET/1.0/req/Core/Tasking/taskDescription**
- **http://www.opengis.net/spec/RESET/1.0/req/Core/Tasking/ taskDescription/content**
- **http://www.opengis.net/spec/RESET/1.0/req/Feasibility/FeasibilityAnalysis/creation/content/status**
- **http://www.opengis.net/spec/RESET/1.0/req/Feasibility/FeasibilityAnalysis/taskDescription/content**

e) Test type: Capability

14.3 Conformance class: Planning

14.3.1 Planning resource

14.3.1.1 POST Planning

a) Test id: http://www.opengis.net/spec/RESET/1.0/conf/ Planning /POST_Planning/

b) Test purpose: To verify whether the **RESET** Server under test is able to manage HTTP **POST** requests on Planning resource (i.e. it is able to create Planning resources).

Note that the request message shall be prepared in order to allow the **RESET** Server under test to accept the task (i.e. it has to be created using GET Tasking request description response in section 14.5.2.2).

c) Test method:

Verify that the **RESET** Server under test accepts the following request:

- HTTP Method: POST
- PATH: http://<hostname>:<port>/<context path>/RESET/1.0.0/planning
- ENTITY: <eosps:TaskingParameters>

Verify the response message:

- HTTP Status: 200 OK
- HTTP Entity Body:
  - complies with the SubmitResponse described in the **EO-SPS** specification [NR22];
  - contains a StatusReport element as defined RESET Planning Requirement Class section 10;
  - StatusReport contains a "PLANNING ACCEPTED" status identifier;
  - contains at least one ProgrammingStatus element as defined in RESET Planning Requirement Class section 10.3.1;
  - Segment structures within the ProgrammingStatus have a "ACCEPTED" status. Some of them can also have the "REJECTED" status.

Pass if the assertion is satisfied; fail otherwise.

d) References:

- **http://www.opengis.net/spec/RESET/1.0/req/ Planning Planning/creation**
- **http://www.opengis.net/spec/RESET/1.0/req/ Planning/Planning/creation/content**
- **http://www.opengis.net/spec/RESET/1.0/req/ Planning/Planning/creation/content/status**
- **http://www.opengis.net/spec/RESET/1.0/req/ Planning/Planning/ creation/content/segments**

e) Test type: Capability

14.3.1.2 GET TaskStatus

a) Test id:

http://www.opengis.net/spec/RESET/1.0/conf/Planning/GET_Planning/taskID/

b) Test purpose: To verify whether the **RESET** Server under test is able to retrieve the status of a previously created Planning resource.

Note that the test must be prepared inserting an existing task id (using the POST Planning operation) and not using an arbitrary value.

c) **Test method**:

Verify that the **RESET** Server under test accepts the following request:

- HTTP Method: GET
- PATH:
  http://<hostname>:<port>/<context path>/RESET/1.0.0/planning/{taskID}

Verify the response message:

- HTTP Status: 200 OK
- HTTP Entity Body:
  - complies with the GetStatusResponse described in the **EO-SPS** specification [NR22];
  - contains a StatusReport element as defined in RESET Planning Requirement Class section 10.3.1;
  - StatusReport contains a status identifier that corresponds to one of the status described in Table 10-1 Planning task status description.

Pass if the assertion is satisfied; fail otherwise.

d) **References**:

- **http://www.opengis.net/spec/RESET/1.0/req/Core/Tasking/taskStatus**
- **http://www.opengis.net/spec/RESET/1.0/req/Feasibility/FeasibilityAnalysis/creation/content/status**
- **http://www.opengis.net/spec/RESET/1.0/req/Planning/Planning/taskResultDescription/content**
- **http://www.opengis.net/spec/RESET/1.0/req/Planning/Planning/taskResultDescription/content**

e) **Test type**: Capability

14.3.1.3 GET TaskDescription

a) **Test id:**

http://www.opengis.net/spec/RESET/1.0/conf/Planning/GET_Planning/taskID/segments/

b) **Test purpose**: To verify whether the **RESET** Server under test is able to retrieve the status and segments of a previously created Planning resource.

Note that the test must be prepared inserting an existing task id (using the POST Planning operation section 14.3.1.1) and not using an arbitrary value.

c) **Test method**:

Verify that the **RESET** Server under test accepts the following request:

- HTTP Method: GET
- PATH:
  http://<hostname>:<port>/<contextpath>/RESET/1.0.0/planning
  /{taskID}/segments

Verify the response message:

- HTTP Status: 200 OK
- HTTP Entity Body:
  - complies with the GetTaskResponse described in the **EO-SPS** specification [NR22];
  - contains a StatusReport element as defined in RESET Planning Requirement Class section 10.3.1;
  - StatusReport contains a status identifier that corresponds to one of the status described in Table 10-1 Planning task status description;
  - contains at least one ProgrammingStatus element as defined in RESET Planning Requirement Class section 10.3.1;
  - Segment structures within the ProgrammingStatus have a status that corresponds to one of the status described in Table 10-2 Planning segments status description.

Pass if the assertion is satisfied; fail otherwise.

d) References:

- **http://www.opengis.net/spec/RESET/1.0/req/Core/Tasking/taskDescription**
- **http://www.opengis.net/spec/RESET/1.0/req/Core/Tasking/taskDescription/content**
- **http://www.opengis.net/spec/RESET/1.0/req/Planning/Planning/ creation/content/status**
- **http://www.opengis.net/spec/RESET/1.0/req/ Planning/Planning/taskDescription/content**

e) Test type: Capability

14.3.1.4 GET TaskResults

a) Test id:

http://www.opengis.net/spec/RESET/1.0/conf/Planning/GET_Planning/taskID/results/

b) Test purpose: To verify whether the **RESET** Server under test is able to retrieve the results produced by a previously created Planning resource.

Note that the test must be prepared inserting an existing task id (using the POST Planning operation in section 14.3.1.1) and not using an arbitrary value.

Note that the test response shall only provide results for ACQUIRED segments. If no segments are ACQUIRED, then the response will contain a DataNotAvailable element.

c) Test method:

Verify that the **RESET** Server under test accepts the following request:

- HTTP Method: GET
- PATH: http://<hostname>:<port>/<context path>/RESET/1.0.0/planning/{taskID}/results

Verify the response message:

- HTTP Status: 200 OK

- HTTP Entity Body:
  - complies with the DescribeResultAccessResponse described in the **EO-SPS** specification [NR22].

Pass if the assertion is satisfied; fail otherwise.

d) References:

- **http://www.opengis.net/spec/RESET/1.0/req/Planning/Planning/taskResultDescription/**
- **http://www.opengis.net/spec/RESET/1.0/req/Planning/Planning/taskResultDescription/content**

e) Test type: Capability

14.3.1.5 POST SegmentValidation

a) Test id:

http://www.opengis.net/spec/RESET/1.0/conf/Planning/GET_Planning/taskID/results/

b) Test purpose: To verify whether the **RESET** Server under test is able to retrieve the results produced by a previously created Planning resource.

Note that the test must be prepared inserting an existing task id (using the POST Planning operation in section 14.3.1.1) and not using an arbitrary value.

Note that the test segments to be validated must be in the ACQUIRED state.

c) Test method:

Verify that the RESET Server under test accepts the following request:

- HTTP Method: POST
- PATH:
  http://<hostname>:<port>/<context path>/RESET/1.0.0/planning/{taskID}/segments/{segmentID*}

Verify the response message:

- HTTP Status: 200 OK
- HTTP Entity Body:
  - complies with the ValidateResponse described in the **EO-SPS** specification [NR22].

Pass if the assertion is satisfied; fail otherwise.

d) References:

- **http://www.opengis.net/spec/RESET/1.0/req/Planning/Planning/submission**
- **http://www.opengis.net/spec/RESET/1.0/req/FeasibilityPlanning/FeasibilityAnalysis/submission/content**
- **http://www.opengis.net/spec/RESET/1.0/req/FeasibilityPlanning/FeasibilityAnalysis/submission/content/status**
- **http://www.opengis.net/spec/RESET/1.0/req/FeasibilityPlanning/FeasibilityAnalysis/submission/content/segments**

e) Test type: Capability

14.4 Conformance class: Feasibility Planning

14.4.1 Feasibility resource

14.4.1.1 POST Feasibility

a) Test id:

http://www.opengis.net/spec/RESET/1.0/conf/FeasibilityPlanning/POST_Feasibility/

b) Test purpose: To verify whether the **RESET** Server under test is able to manage HTTP **POST** requests on Feasibility resource (i.e. it is able to create Feasibility resources).

Note that the request message shall be prepared in order to allow the **RESET** Server under test to accept the task (i.e. it has to be created using GET Tasking request description response section 9.3.2 above).

c) Test method:

Verify that the RESET Server under test accepts the following request:

- HTTP Method: POST
- PATH: http://<hostname>:<port>/<context path>/RESET/1.0.0/feasibility
- ENTITY: <eosps:TaskingParameters>

Verify the response message:

- HTTP Status: 200 OK
- HTTP Entity Body:
  - complies with the GetFeasibilityResponse described in the **EO-SPS** specification [NR22];
  - contains a StatusReport element as defined in RESET Feasibility Requirement Clas section 9 aboves;
  - StatusReport contains a “FEASIBILITY COMPLETED” status identifier;
  - contains at least one FeasibilityStudy element as defined in RESET Feasibility Requirement Class section 9 above;
  - Segment structures within the FeasibilityStudy have a “POTENTIAL” status.

Pass if the assertion is satisfied; fail otherwise.

d) References:

- **http://www.opengis.net/spec/RESET/1.0/req/Feasibility/FeasibilityAnalysis/creation**
- **http://www.opengis.net/spec/RESET/1.0/req/Feasibility/FeasibilityAnalysis/creation/content**
- **http://www.opengis.net/spec/RESET/1.0/req/Feasibility/FeasibilityAnalysis/creation/content/status**
- **http://www.opengis.net/spec/RESET/1.0/req/Feasibility/FeasibilityAnalysis/creation/content/segments**

e) Test type: Capability

14.4.1.2 POST FeasibilityTask

a) Test id:

http://www.opengis.net/spec/RESET/1.0/conf/FeasibilityPlanning/POST_Feasibility/taskID/

b) Test purpose: To verify whether the **RESET** Server under test is able to manage HTTP **POST** requests on a previously created Feasibility resource (i.e. it is able to submit a Feasibility resource in order to create an associated Planning resource).

c) Note that the request message shall be prepared in order to allow the RESET Server under test to accept the task (i.e. it has to contain the task identifier (taskID) in the path, as well as a list of segments provided through the segment identifier (segmentID) as defined in the RESET Feasibility Planning Requirement Class section 10 above).

Note that the test must be prepared inserting an existing task id (using the POST Feasibility operation section 9 above) and not using an arbitrary value.

d) Test method:

Verify that the **RESET** Server under test accepts the following request:

- HTTP Method: POST
- PATH:
  http://<hostname>:<port>/<context path>/RESET/1.0.0/feasibility/{taskID}/segments/{segmentID*}
- ENTITY:none

Verify the response message:

- HTTP Status: 200 OK
- HTTP Entity Body:
  - complies with the SubmitResponse described in the **EO-SPS** specification [NR22];
  - contains a StatusReport element as defined RESET Planning Requirement Class section 10 above ;
  - StatusReport contains a "PLANNING ACCEPTED" status identifier;
  - contains at least one ProgrammingStatus element as defined in RESET Planning Requirement Class section 10 above;
  - Segment structures within the ProgrammingStatus have a "ACCEPTED" status. Some of them can also have the "REJECTED" status.

Pass if the assertion is satisfied; fail otherwise.

e) References:

- **http://www.opengis.net/spec/RESET/1.0/req/FeasibilityPlanning/Feasibility/submission**
- **http://www.opengis.net/spec/RESET/1.0/req/FeasibilityPlanning/Feasibility/submission /content**
- **http://www.opengis.net/spec/RESET/1.0/req/FeasibilityPlanning/Feasibility/submission/status**
- **http://www.opengis.net/spec/RESET/1.0/req/FeasibilityPlanning/FeasibilityAnalysis/submission/content/segments**

f) Test type: Capability

14.4.2 Planning resource

14.4.2.1 GET TaskStatus

a) Test id:

http://www.opengis.net/spec/RESET/1.0/conf/FeasibilityPlanning/GET_Planning/taskID/

b) Test purpose: To verify whether the **RESET** Server under test is able to retrieve the status of a Planning resource previously created by a feasibility submission.

Note that the test must be prepared inserting an existing task id (using the POST FeasibilityTask operation section 9 above) and not using an arbitrary value.

c) Test method:

Verify that the **RESET** Server under test accepts the following request:

- HTTP Method: GET
- PATH:
  http://<hostname>:<port>/<context path>/RESET/1.0.0/planning/{taskID}

Verify the response message:

- HTTP Status: 200 OK
- HTTP Entity Body:
  - complies with the GetStatusResponse described in the **EO-SPS** specification [NR22];
  - contains a StatusReport element as defined in RESET Planning Requirement Class section 10 above;
  - StatusReport contains a status identifier that corresponds to one of the status described in Table 10-1 Planning task status description.

Pass if the assertion is satisfied; fail otherwise.

d) References:

- **http://www.opengis.net/spec/RESET/1.0/req/Core/Tasking/taskStatus**
- **http://www.opengis.net/spec/RESET/1.0/req/Feasibility/FeasibilityAnalysis/creation/content/status**
- **http://www.opengis.net/spec/RESET/1.0/req/Planning/Planning/taskResultDescription/content**
- **http://www.opengis.net/spec/RESET/1.0/req/Planning/Planning/taskResultDescription/content**

e) Test type: Capability

14.4.2.2 GET TaskDescription

a) Test id:

http://www.opengis.net/spec/RESET/1.0/conf/FeasibilityPlanning/GET_Planning/taskID/segments/

b) Test purpose: To verify whether the **RESET** Server under test is able to retrieve the status and segments of a Planning resource previously created by a feasibility submission.

Note that the test must be prepared inserting an existing task id (using the POST FeasibilityTask operation section 9 above) and not using an arbitrary value.

c) Test method:

Verify that the **RESET** Server under test accepts the following request:

- HTTP Method: GET
- PATH: http://<hostname>:<port>/<context path>/RESET/1.0.0/ planning /{taskID}/segments

Verify the response message:

- HTTP Status: 200 OK
- HTTP Entity Body:
  - complies with the GetTaskResponse described in the **EO-SPS** specification [NR22];
  - contains a StatusReport element as defined in RESET Planning Requirement Class section 10 above;
  - StatusReport contains a status identifier that corresponds to one of the status described in Table 10-1 Planning task status description;
  - contains at least one ProgrammingStatus element as defined in RESET Planning Requirement Class section 10 above;
  - Segment structures within the ProgrammingStatus have a status that corresponds to one of the status described in Table 10-2 Planning segments status description.

Pass if the assertion is satisfied; fail otherwise.

d) References:

- **http://www.opengis.net/spec/RESET/1.0/req/Core/Tasking/taskDescription**
- **http://www.opengis.net/spec/RESET/1.0/req/Core/Tasking/taskDescription/content**
- **http://www.opengis.net/spec/RESET/1.0/req/Planning/Planning/ creation/content/status**
- **http://www.opengis.net/spec/RESET/1.0/req/ Planning/Planning/taskDescription/content**

e) Test type: Capability

14.4.2.3 GET TaskResults

a) Test id:

http://www.opengis.net/spec/RESET/1.0/conf/FeasibilityPlanning/GET_Planning/taskID/results/

b) Test purpose: To verify whether the **RESET** Server under test is able to retrieve the results produced by a Planning resource previously created by a feasibility submission.

Note that the test must be prepared inserting an existing task id (using the POST FeasibilityTask operation section 9 above) and not using an arbitrary value.

Note that the test response shall only provide results for ACQUIRED segments. If no segments are ACQUIRED, then the response will contain a DataNotAvailable element.

c) **Test method**:

Verify that the **RESET** Server under test accepts the following request:

- HTTP Method: GET
- PATH:
  http://<hostname>:<port>/<context path>/RESET/1.0.0/planning/{taskID}/results

Verify the response message:

- HTTP Status: 200 OK
- HTTP Entity Body:
  - complies with the DescribeResultAccessResponse described in the **EO-SPS** specification [NR22].

Pass if the assertion is satisfied; fail otherwise.

d) **References**:

- **http://www.opengis.net/spec/RESET/1.0/req/Planning/Planning/taskResultDescription/**
- **http://www.opengis.net/spec/RESET/1.0/req/Planning/Planning/taskResultDescription/content**

e) **Test type**: Capability

14.5 Conformance class: Reservation

14.5.1 Capabilities resource

14.5.1.1 GET Capabilities/OperationsMetadata

a) Test id:

http://www.opengis.net/spec/RESET/1.0/conf/Reservation/GET_Capabilities/OperationsMetadata

b) Test purpose: To verify that the **RESET** Server under test correctly returns the OperationsMetadata section of a Capabilities resource and that this structure contains the description of the optional operations necessary for the Reservation process.

c) Test method:

Verify that the **RESET** Server under test accepts the following request:

- HTTP Method: GET
- PATH: http://<hostname>:<port>/<context path>/RESET/1.0.0/Contents

Verify the response message:

- HTTP Status: 200 OK
- HTTP Entity Body:
  - complies with the Contents element of reset.xsd schema
  - at least the operations of the Core and Planning classes is implemented;
  - the operations of the Reservation class(Reserve, Update, Confirm) are implemented.

Pass if the assertion is satisfied; fail otherwise.

d) References:

- **http://www.opengis.net/spec/RESET/1.0/req/Core/Capabilities/Contents**
- **http://www.opengis.net/spec/RESET/1.0/req/Core/Capabilities/resource/resourceURL**
- **http://www.opengis.net/spec/RESET/1.0/req/Core/Capabilities/resource/classe**

e) Test type: Capability

14.5.2 Planning resource

14.5.2.1 GET TaskStatus

a) Test id:

http://www.opengis.net/spec/RESET/1.0/conf/Reservation/GET_Planning/taskID/

b) Test purpose: To verify whether the **RESET** Server under test is able to retrieve the status of a Planning resource previously created by a reservation confirmation.

Note that the test must be prepared inserting an existing task id (using the POST Reservation Task operation section 12 above) and not using an arbitrary value.

c) Test method:

Verify that the **RESET** Server under test accepts the following request:

- HTTP Method: GET
- PATH: http://<hostname>:<port>/<context path>/RESET/1.0.0/planning/{taskID}

Verify the response message:

- HTTP Status: 200 OK
- HTTP Entity Body:
  - complies with the GetStatusResponse described in the **EO-SPS** specification [NR22];
  - contains a StatusReport element as defined in RESET Planning Requirement Class section 10 above;
  - StatusReport contains a status identifier that corresponds to one of the status described in Table 10-1 Planning task status description.

Pass if the assertion is satisfied; fail otherwise.

d) References:

- **http://www.opengis.net/spec/RESET/1.0/req/Core/Tasking/taskStatus**
- **http://www.opengis.net/spec/RESET/1.0/req/Feasibility/FeasibilityAnalysis/creation/content/status**
- **http://www.opengis.net/spec/RESET/1.0/req/Planning/Planning/taskResultDescription/content**
- **http://www.opengis.net/spec/RESET/1.0/req/Planning/Planning/taskResultDescription/content**

e) Test type: Capability

14.5.2.2 GET TaskDescription

a) Test id:

http://www.opengis.net/spec/RESET/1.0/conf/Reservation/GET_Planning/taskID/segments/

b) Test purpose: To verify whether the **RESET** Server under test is able to retrieve the status and segments of a Planning resource previously created by a reservation confirmation.

Note that the test must be prepared inserting an existing task id (using the POST Reservation Task operation section 10 above) and not using an arbitrary value.

c) Test method:

Verify that the **RESET** Server under test accepts the following request:

- HTTP Method: GET
- PATH: http://<hostname>:<port>/<context path>/RESET/1.0.0/ planning /{taskID}/segments

Verify the response message:

- HTTP Status: 200 OK
- HTTP Entity Body:

- complies with the GetTaskResponse described in the **EO-SPS** specification [NR22];
- contains a StatusReport element as defined in RESET Planning Requirement Class section 10 above;
- StatusReport contains a status identifier that corresponds to one of the status described in Table 10-1 Planning task status description;
- contains at least one ProgrammingStatus element as defined in RESET Planning Requirement Class section 10 above;
- Segment structures within the ProgrammingStatus have a status that corresponds to one of the status described in Table 10-2 Planning segments status description.

Pass if the assertion is satisfied; fail otherwise.

d) References:

- **http://www.opengis.net/spec/RESET/1.0/req/Core/Tasking/taskDescription**
- **http://www.opengis.net/spec/RESET/1.0/req/Core/Tasking/taskDescription/content**
- **http://www.opengis.net/spec/RESET/1.0/req/Planning/Planning/ creation/content/status**
- **http://www.opengis.net/spec/RESET/1.0/req/ Planning/Planning/taskDescription/content**

e) Test type: Capability

14.5.2.3 GET TaskResults

a) Test id:

http://www.opengis.net/spec/RESET/1.0/conf/Reservation/GET_Planning/taskID/results/

b) Test purpose: To verify whether the **RESET** Server under test is able to retrieve the results produced by a Planning resource previously created by a reservation confirmation.

Note that the test must be prepared inserting an existing task id (using the POST Reservation Task operation section 10 above) and not using an arbitrary value.

Note that the test response shall only provide results for ACQUIRED segments. If no segments are ACQUIRED, then the response will contain a DataNotAvailable element.

c) Test method:

Verify that the **RESET** Server under test accepts the following request:

- HTTP Method: GET
- PATH: http://<hostname>:<port>/<context path>/RESET/1.0.0/planning/{taskID}/results

Verify the response message:

- HTTP Status: 200 OK
- HTTP Entity Body:
  - complies with the DescribeResultAccessResponse described in the **EO-SPS** specification [NR22].

Pass if the assertion is satisfied; fail otherwise.

d) **References**:

- http://www.opengis.net/spec/RESET/1.0/req/Planning/Planning/taskResultDescription/
- http://www.opengis.net/spec/RESET/1.0/req/Planning/Planning/taskResultDescription/content

e) **Test type**: Capability

14.5.3 Reservation resource

14.5.3.1 POST Reservation

a) **Test id:**

http://www.opengis.net/spec/RESET/1.0/conf/Reservation/POST_Reservation/

b) **Test purpose**: To verify whether the **RESET** Server under test is able to manage HTTP **POST** requests on Reservation resource (i.e. it is able to create Reservation resources).

Note that the request message shall be prepared in order to allow the RESET Server under test to accept the task (i.e. it has to be created using GET Tasking request description response in 8.3.2 above).

c) **Test method**:

Verify that the **RESET** Server under test accepts the following request:

- HTTP Method: POST
- PATH: http://<hostname>:<port>/<context path>/RESET/1.0.0/reservation
- ENTITY: <eosps:TaskingParameters>

Verify the response message:

- HTTP Status: 200 OK
- HTTP Entity Body:
  - complies with the ReserveResponse described in the **EO-SPS** specification [NR22];
  - contains a StatusReport element as defined in RESET Reservation Requirement Class section 10 above;
  - StatusReport contains a "PLANNING RESERVED" status identifier;
  - contains at least one ProgrammingStatus element as defined in in RESET Reservation Requirement Class;
  - Segment structures within the ProgrammingStatus have a "POTENTIAL" status section 10 above.

Pass if the assertion is satisfied; fail otherwise.

d) **References**:

- http://www.opengis.net/spec/RESET/1.0/req/Reservation/ Reservation / creation
- http://www.opengis.net/spec/RESET/1.0/req/Reservation/Reservation/creation/content
- http://www.opengis.net/spec/RESET/1.0/req/Reservation/Reservation/creation/content/status
- http://www.opengis.net/spec/RESET/1.0/req /Reservation/Reservation/creation/content/segments

e) **Test type**: Capability

14.5.3.2 PUT Reservation Task

a) Test id:

http://www.opengis.net/spec/RESET/1.0/conf/Reservation/POST_Reservation/

b) Test purpose: To verify whether the **RESET** Server under test is able to manage HTTP **PUT** requests on Reservation resource (i.e. it is able to update a Reservation resource).

Note that the request message shall be prepared in order to allow the **RESET** Server under test to accept the task (i.e. it has to be created using GET Tasking request description response in 8.3.2 above).

c) Test method:

Verify that the **RESET** Server under test accepts the following request:

- HTTP Method: PUT
- PATH:
  http://<hostname>:<port>/<context path>/RESET/1.0.0/reservation/{taskID}
- ENTITY: <eosps:TaskingParameters>
  <sps: reservationExpiration>

Verify the response message:

- HTTP Status: 200 OK
- HTTP Entity Body:
  - complies with the UpdateResponse described in the **EO-SPS** specification [NR22];
  - contains a StatusReport element as defined in RESET Reservation Requirement Class section 10 above;
  - StatusReport contains a "PLANNING RESERVED" status identifier;
  - contains at least one ProgrammingStatus element as defined in the RESET Reservation Requirement Class section 10 above;
  - Segment structures within the ProgrammingStatus have a "POTENTIAL" status;
  - Contains a <sps:reservationExpiration> element as defined in the RESET Reservation Requirement Class section 10 above.

Pass if the assertion is satisfied; fail otherwise.

d) References:

- **http://www.opengis.net/spec/RESET/1.0/req/Reservation/Reservation/modification**
- **http://www.opengis.net/spec/RESET/1.0/req/Reservation/Reservation/modification/content**

e) Test type: Capability

14.5.3.3 POST Reservation Task

a) **Test id:**

http://www.opengis.net/spec/RESET/1.0/conf/Reservation/POST_Reservation/

b) **Test purpose**: To verify whether the **RESET** Server under test is able to manage HTTP **POST** requests on Reservation resource (i.e. it is able to submit a Reservation resource).

c) **Test method**:

Verify that the **RESET** Server under test accepts the following request:

- HTTP Method: POST
- PATH:
  http://<hostname>:<port>/<context path>/RESET/1.0.0/reservation/{taskID}
- ENTITY: none

Verify the response message:

- HTTP Status: 200 OK
- HTTP Entity Body:
  - complies with the ConfirmResponse described in the **EO-SPS** specification [NR22];
  - contains a StatusReport element as defined RESET Planning Requirement Class section 10 above;
  - StatusReport contains a "PLANNING ACCEPTED" status identifier;
  - contains at least one ProgrammingStatus element as defined in RESET Planning Requirement Class section 10 above;
  - Segment structures within the ProgrammingStatus have a "ACCEPTED" status. Some of them can also have the "REJECTED" status.

Pass if the assertion is satisfied; fail otherwise.

d) **References**:

- **http://www.opengis.net/spec/RESET/1.0/req/Reservation/Reservation/submission**
- **http://www.opengis.net/spec/RESET/1.0/req/Reservation/Reservation/submission/content**
- **http://www.opengis.net/spec/RESET/1.0/req/Reservation/Reservation/submission/content/status**
- **http://www.opengis.net/spec/RESET/1.0/req /Planning/ Planning /creation/content/segments**

e) **Test type**: Capability

14.6 Conformance class: Cancellation

14.6.1 Capabilities resource

a) Test id:

http://www.opengis.net/spec/RESET/1.0/conf/Cancellation/GET_Capabilities/OperationsMetadata

b) Test purpose: To verify that the **RESET** Server under test correctly returns the Resource section of a Capabilities resource and that this structure contains the description of the optional operations necessary for the Cancellation process.

c) Test method:

Verify that the **RESET** Server under test accepts the following request:

- HTTP Method: GET
- PATH: http://<hostname>:<port>/<context path>/RESET/1.0.0/resourceURL/

Verify the response message:

- HTTP Status: 200 OK
- HTTP Entity Body:
  - complies with the Contents element of reset.xsd schema
  - at least the operations of the Core and Planning classes is implemented;
  - the operation of the Cancellation class(Cancel) are implemented.

Pass if the assertion is satisfied; fail otherwise.

d) References:

- **http://www.opengis.net/spec/RESET/1.0/req/Core/Capabilities/Contents**
- **http://www.opengis.net/spec/RESET/1.0/req/Core/Capabilities/resource/resourceURL**
- **http://www.opengis.net/spec/RESET/1.0/req/Core/Capabilities/resource/classe**

e) Test type: Capability

14.6.2 Planning resource

14.6.2.1 DELETE Planning Task

a) Test id:

http://www.opengis.net/spec/RESET/1.0/conf/Cancellation/DELETE_Planning/

b) Test purpose: To verify whether the **RESET** Server under test is able to manage HTTP **DELETE** requests on a Planning resource (i.e. it is able to cancel a Planning resource).

Note that the test must be prepared inserting an existing task id (using the POST Planning operation section 14.3.1 above) and not using an arbitrary value.

c) **Test method**:

Verify that the **RESET** Server under test accepts the following request:

- HTTP Method: DELETE
- PATH: http://<hostname>:<port>/<context path>/RESET/1.0.0/planning/{taskID}
- ENTITY: none

Verify the response message:

- HTTP Status: 200 OK
- HTTP Entity Body:
  - complies with the CancelResponse described in the **EO-SPS** specification [NR22];
  - contains a StatusReport element as defined in RESET Cancellation Requirement Class section 13 above;
  - StatusReport contains a "PLANNING CANCELLED" status identifier;
  - contains at least one ProgrammingStatus element as defined in RESET Cancellation Requirement Class section 13 above;
  - Segment structures within the ProgrammingStatus have a "CANCELLED" status, unless they have been acquired or rejected before cancellation.

Pass if the assertion is satisfied; fail otherwise.

d) **References**:

- **http://www.opengis.net/spec/RESET/1.0/req/Cancellation/TaskCancellation**
- **http://www.opengis.net/spec/RESET/1.0/req/Cancellation/TaskCancellation/content**
- **http://www.opengis.net/spec/RESET/1.0/req/Cancellation/TaskCancellation/content/status**
- **http://www.opengis.net/spec/RESET/1.0/req/Cancellation/TaskCancellation/content/status/segments**

e) **Test type**: Capability

14.6.3 Reservation resource

14.6.3.1 DELETE Reservation Task

a) **Test id:**

http://www.opengis.net/spec/RESET/1.0/conf/Cancellation/DELETE_Reservation/

b) **Test purpose**: To verify whether the **RESET** Server under test is able to manage HTTP **POST** requests on a Reservation resource (i.e. it is able to cancel a Reservation resource).

Note that the test must be prepared inserting an existing task id (using the POST Reservation operation section 14.5.1 above13 above) and not using an arbitrary value.

c) Test method:

Verify that the **RESET** Server under test accepts the following request:

- HTTP Method: DELETE
- PATH: http://<hostname>:<port>/<context path>/RESET/1.0.0/reservation{taskID}
- ENTITY: none

Verify the response message:

- HTTP Status: 200 OK
- HTTP Entity Body:
  - complies with the CancelResponse described in the **EO-SPS** specification [NR22];
  - contains a StatusReport element as defined in RESET Cancellation Requirement Class section 13 above;
  - StatusReport contains a "PLANNING CANCELLED" status identifier;
  - contains at least one ProgrammingStatus element as defined in in RESET Cancellation Requirement Class section 13 above;
  - Segment structures within the ProgrammingStatus have a "CANCELLED" status, unless they have been acquired or rejected before cancellation.

Pass if the assertion is satisfied; fail otherwise.

d) References:

- **http://www.opengis.net/spec/RESET/1.0/req/Cancellation/TaskCancellation**
- **http://www.opengis.net/spec/RESET/1.0/req/Cancellation/TaskCancellation/content**
- **http://www.opengis.net/spec/RESET/1.0/req/Cancellation/TaskCancellation/content/status**
- **http://www.opengis.net/spec/RESET/1.0/req/Cancellation/TaskCancellation/content/status/segments**

e) Test type: Capability

15. Annex B: Revision history

Date	Release	Author	Paragraph modified	Description
2014/28/01	0.0.1	Nicolas Fanjeau Sebastian Ulrich	All	First release of the document