con terra GmbH

# Ordering Services for Earth Observation Products - Abstract Test Suite

ESA Heterogeneous Missions Accessibility FollowOn - Task 4
Order (**CONTRACT N. ESA/ESRIN 22508/09/I-LG)**

| | |
|---|---|
| Title | Ordering Services for Earth Observation Products - Abstract Test Suite |
| Creator | Dr. Uwe Voges (con terra, Germany) |
| Issued | 17/12/2009 16:36:00 |
| Modified | 25/01/2010 |
| Subject | HMA OGC Order Services |
| Publisher | con terra GmbH |
| Type | Abstract Test Suite |
| isReferencedBy | ESA HMA |
| Abstract | This document describes the Abstract Test Suite (ATS) of the OGC Ordering Services for Earth Observation Products. An ATS provides a basis for developing an executable test suite (ETS). . |
| Contributors | Daniele Marchionni (ELSAG DATAMAT, Italy) |
| Format | MS-Word 2003 (doc) |
| Rights | ESA |
| bibliographicCitation | |
| Language | Eng |
| References | |
| Coverage | World |

Dublin Core Metadata Elements (s. http://www.dublincore.org)

# History

| Version | Status[2] | Datum | Author(s) | Explanation |
|---|---|---|---|---|
| 0.1 | Inprogress | 17/12/2009 | Uwe Voges | Initialization and inclusion of first draft of Abstract Test Suite from temporary document |
| 0.2 | Inprogress | 22/12/2009 | Uwe Voges | Solved comments from Daniele and added different test cases |
| 0.3 | Inprogress | 15/01/2009 | Uwe Voges | Integrated asycnch behaviour, Changed GetStatus behaviour |
| 0.4 | Inprogress | 25/01/2009 | Uwe Voges | Improved descriptions of asynch behaviour. Solved review comments from DM. Changed optional Conformance Classes for asynchronous behaviour into Conformance Classes. |
|  |  |  |  |  |
|  |  |  |  |  |

[2] in progress, final, reviewed

# Content

# Figures

# 1 Introduction

## 1.1 Purpose and Scope

This document describes the Abstract Test Suite (ATS) of the OGC Ordering Services for Earth Observation Products. 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.

The assertions are gleaned from a set of specification documents; the dependencies among these specifications are shown in the figure below where each specification is represented as a UML package.

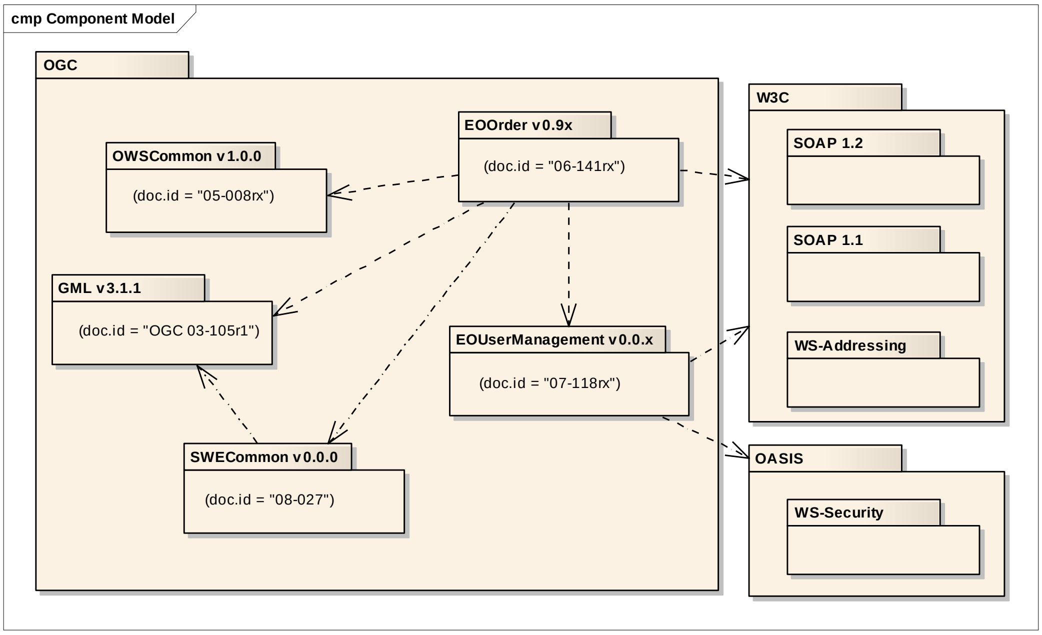

*Figure 1: Package Dependency of EO Order Specification*

## 1.2 Applicable and Reference Documents

[AD1] ECSS – Space Engineering Standards – Software ECSS-E-ST-40C, 6 March 2009.

[AD-02] OGC™ Catalogue Services Specification 2.0 Extension Package for ebRIM (ISO/TS 15000-3) Application Profile: Earth Observation Products OGC 06-131r6 0.2.407 May 2008

[AD-03] Application schema for Earth Observation products OGC 06-080r2 0.9.3 21 Jul 2008

[AD-04] Ordering Services for Earth Observation Products OGC 06-141r2 0.9.5 draft 1 13 Nov 2009

[AD-05] Proposal for HMA Follow On Task 4 – Order 1.013 Mar 2009

[AD-06] User Management Interfaces for Earth Observation Services OGC 07-118r1 0.0.4 30 Jun 2009

## 1.3 Definition and Acronyms

| | |
|---|---|
| AR | Acceptance Review |
| CDR | Critical Design Review |
| DAIL | Data Access Integration Layer |
| DDF | Design Definition File |
| DJF | Design Justification File |
| EO | Earth Observation |
| ESA | European Space Agency |
| FP | Final Presentation |
| HMA | Heterogeneous Missions Accessibility |
| INSPIRE | Infrastructure for Spatial Information in Europe |
| ICD | Interface Control Document |

| | |
|---|---|
| KO | Kick off meeting |
| OGC | Open Geospatial Consortium |
| OWL | Web Ontology Language |
| PDR | Preliminary Design Review |
| RDF | Resource Description Framework |
| SDD | Software Design Document |
| SMAAD | Semantic-web Mediated Access Across Domains |
| SRS | Software Requirements Specification |
| SSE | Service Support Environment |
| TS | Technical Specification |
| UML | Unified Modelling Language |

## 1.1 UML Notations

### 1.1.1 UML Class Diagrams

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

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

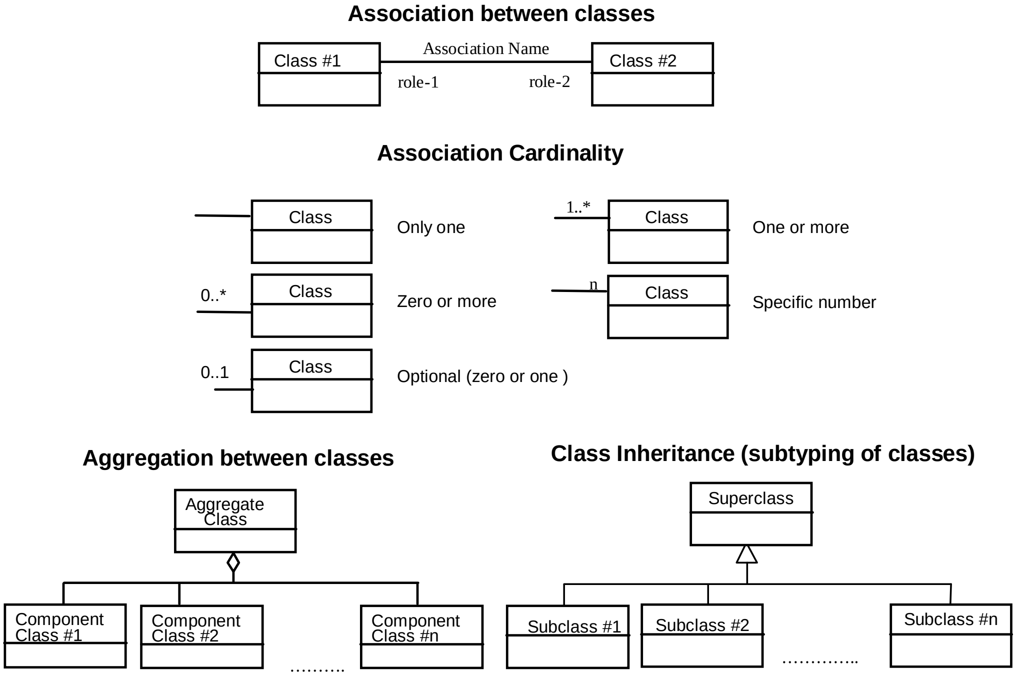

*Figure 2: UML Class Diagram Notations*

The following stereotypes of UML classes are often used:

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

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

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

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

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

### 1.1.2 UML Component Diagrams

Component diagrams are similar to class diagrams but concentrate on higher, sub-system level abstractions. Component diagrams typically contain packages, components, interfaces and their relationships.

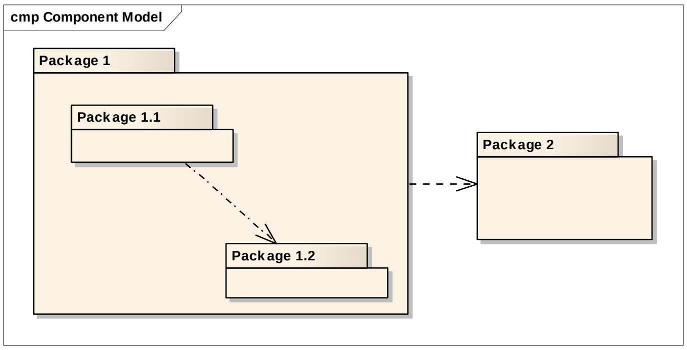

*Figure 3: UML Component Diagram Notations*

### 1.1.3 UML State Diagrams

State diagrams are used to detail the transitions or changes of state an object can go through in the system. They show how an object moves from one state to another and the rules that govern that change. State diagrams typically have a start and end condition.

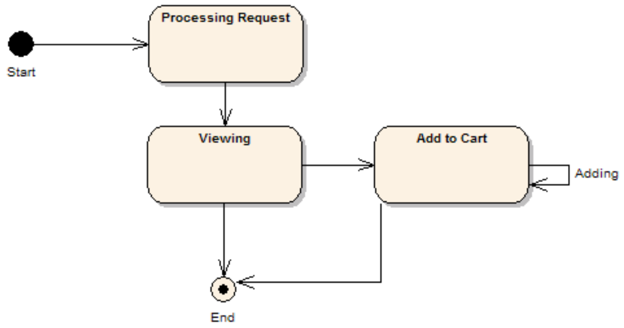

*Figure 4: UML State Diagram Notations*

# 2 Conformance

## 2.1 Overview of the Conformance Classes

According to the OGC approach a general UML model of the EO Order Service Interfaces, in the form of a class diagram is shown in Figure 5. This model shows the EO Order Service class plus five other classes with which that class is associated. An EO Order Service is a realization of an OGC Service. Each instance of the EO Order Service class is associated with two or more of these other classes, depending on the abilities included in that service instance. Each of these other classes defines one or several related operations that can be included in an EO Order Service class instance.

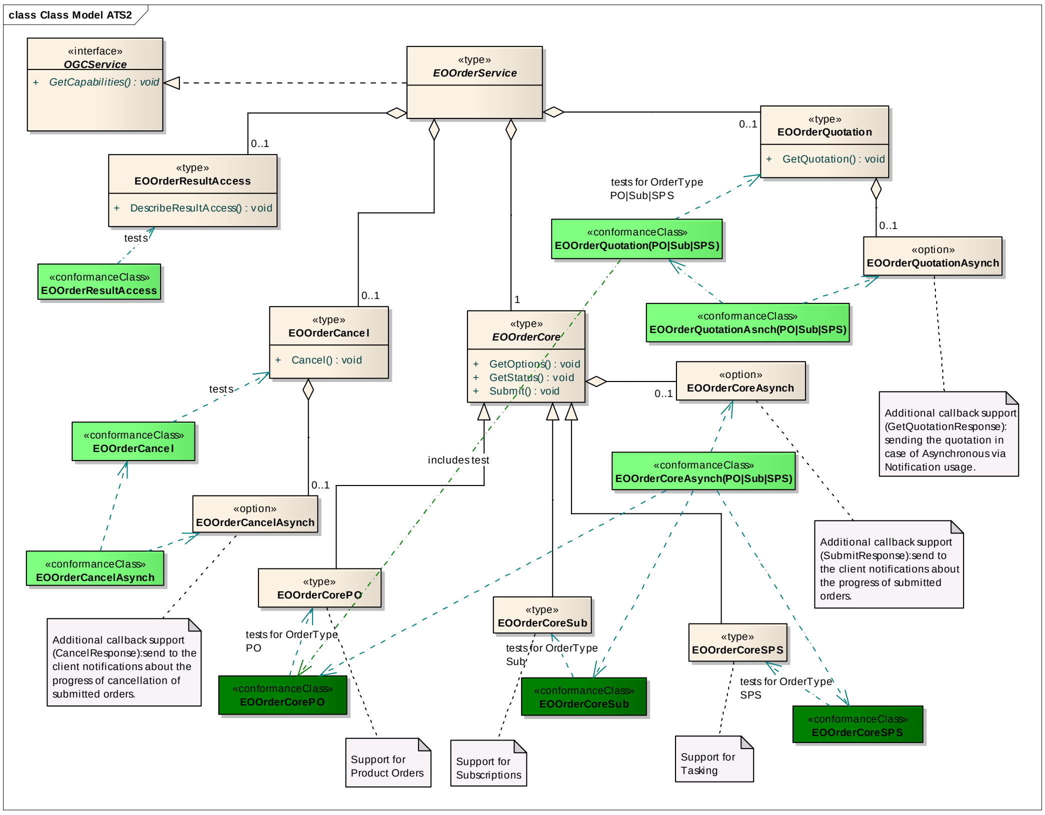

*Figure 5: EO Order Specification interfaces (types) and associated Conformance Classes*

The EO Order Core type is itself an abstract class from which three concrete classes are derived. Each of these three classes does not provide new functionality but is related to a different type of order which can be handled through the operations of the base class.

In Figure 5 an instance of the EO Order Service type is a composite object that is a high-level characterization of an EO Order Service. Its constituent objects are themselves components that provide functional behaviours to address particular areas of concern. The associated classes shown in this figure are mandatory or optional for implementation as indicated by the association multiplicity in the UML diagram. Therefore, a compliant EO Order Service shall implement the OGC Service and EO Order Core classes. An EO Order Server can implement additional classes associated with the EO Order Service class. An EO Order implementation shall recognize all operations defined within each included class, and shall generate a message indicating when a particular operation is not implemented.

The five classes that can be associated with the EO Order Service class allow different OGC EO Order Services to provide significantly different abilities.

Each concrete type has one or more Conformance Classes associated (green coloured in Figure 5 and assigned with a green dashed arrow to the type) which are defined within the Abstract Test Suite (ATS). The relatively large number of Conformance Classes result from a three dimensional space of attributes from which the Conformance Classes are defined. The three axis of this space are:

- The **Operations**: a set of mandatory (Core) operations GetCapabilities, GetOptions, Submit and GetStatus can be combined with the three optional operations (GetQuotation, Cancel, DescribeResultAccess)
- The **OrderType** (ProductOrder(PO), Subscription (Sub), Tasking (SPS))
- The **Processing** type (sybchronous, asynchronous with notification)

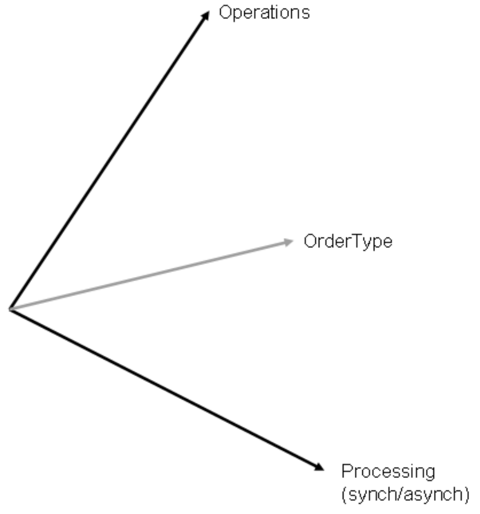

*Figure 6: Three axes defining the space of Conformance Classes*

Each of the following three conformance classes (“dark green” colored in Figure 5).

- *EOOrderCorePO*

- *EOOrderCoreSub*
- *EOOrderCoreSPS*

address the core set of synchronous operations for one specific OrderType. Supporting one or more of these conformance classes is mandatory for an HMA EO Order Server.

The following additional conformance classes test additional optional behaviour ("light green" coloured in the figure above). Support of these conformance classes is optional for an HMA server.

- *EOOrderQuotationPO*
- *EOOrderQuotationSub*
- *EOOrderQuotationSPS*
- *EOOrderCoreAsynchPO*
- *EOOrderCoreAsynchSub*
- *EOOrderCoreAsynchSPS*
- *EOOrderQuotationAsynchPO*
- *EOOrderQuotationAsynchSub*
- *EOOrderQuotationAsynchSPS*
- *EOOrderResultAccess*
- *EOOrderCancel*

The last two options are independent from an OrderType, they require only an orderId (and not an orderSpecificationInformation) as input parameter.

The abbreviation "Asynch" in the names mark Conformance Classes which support additional asynchronous (with notification) behaviour.

Some of the operations within a conformance class are dependant on the procedure/results of other operations within the conformance class. The reason is that sometimes a predefined order of tests is required to be able to test specific capabilities. From this point of view a test class can also be seen as a ***Scenario***. The following UML state diagram visualizes the test cases or scenarios as transitions (from the servers point of view). The EOOrderQuotation is visualized in a separate state diagram below. It should be considered that not all transitions for invalid requests or responses are visualized. The test cases / transitions of the core test classes (*EOOrderCore(PO|Sub|SPS)*, lines coloured green in the state diagram below) cover support for the basic synchronous Order operations.

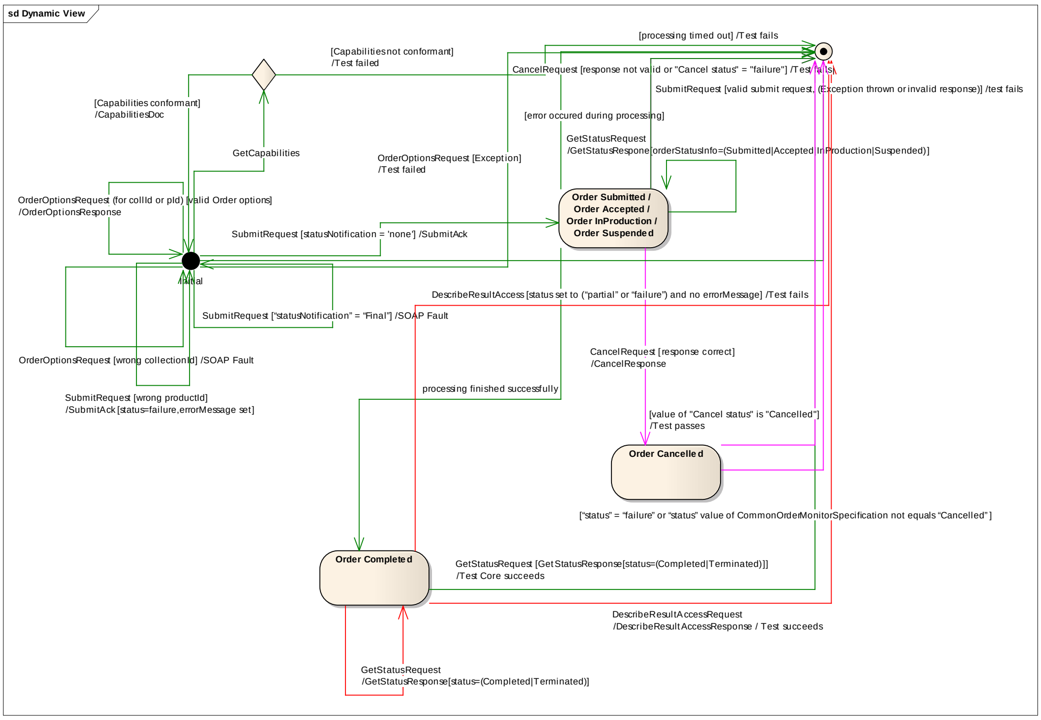
sd Dynamic View
[processing timed out] /Test fails
[Capabilities not conformant]
/Test failed
CancelRequest [response not valid or "Cancel status" = "failure"] /Test fails
SubmitRequest [valid submit request, (Exception thrown or invalid response)] /test fails
[Capabilities conformant]
/CapabilitiesDoc
[error occured during processing]
GetStatusRequest
/GetStatusRespone [orderStatusInfo=(Submitted|Accepted|InProduction|Suspended)]
GetCapabilities
OrderOptionsRequest [Exception]
/Test failed
OrderOptionsRequest (for collId or pId) [valid Order options]
/OrderOptionsResponse
Order Submitted /
Order Accepted /
Order InProduction /
Order Suspended
SubmitRequest [statusNotification = 'none'] /SubmitAck
Initial
DescribeResultAccess [status set to ("partial" or "failure") and no errorMessage] /Test fails
SubmitRequest ["statusNotification" = "Final"] /SOAP Fault
CancelRequest [response correct]
/CancelResponse
OrderOptionsRequest [wrong collectionId] /SOAP Fault
processing finished successfully
SubmitRequest [wrong productId]
/SubmitAck [status=failure,errorMessage set]
[value of "Cancel status" is "Cancelled"]
/Test passes
Order Cancelled
["status" = "failure" or "status" value of CommonOrderMonitorSpecification not equals "Cancelled"]
Order Completed
GetStatusRequest [GetStatusResponse[status=(Completed|Terminated)]]
/Test Core succeeds
DescribeResultAccessRequest
/DescribeResultAccessResponse / Test succeeds
GetStatusRequest
/GetStatusResponse[status=(Completed|Terminated)]

*Figure 7: EO Order State Diagram (synchronous operations without GetQuotation)*

The test cases (transitions) of the EOOrderCancel Test Class are visualized with pink lines and the test cases (transitions) of the EOOrderResultAccess Test Class are visualized with red lines.

The following UML state diagram visualizes the test cases as transisitions (blue lines) which are covered by the Test Class EOOrderQuotation. You must consider that not all transitions for invalid requests or responses are visualized.

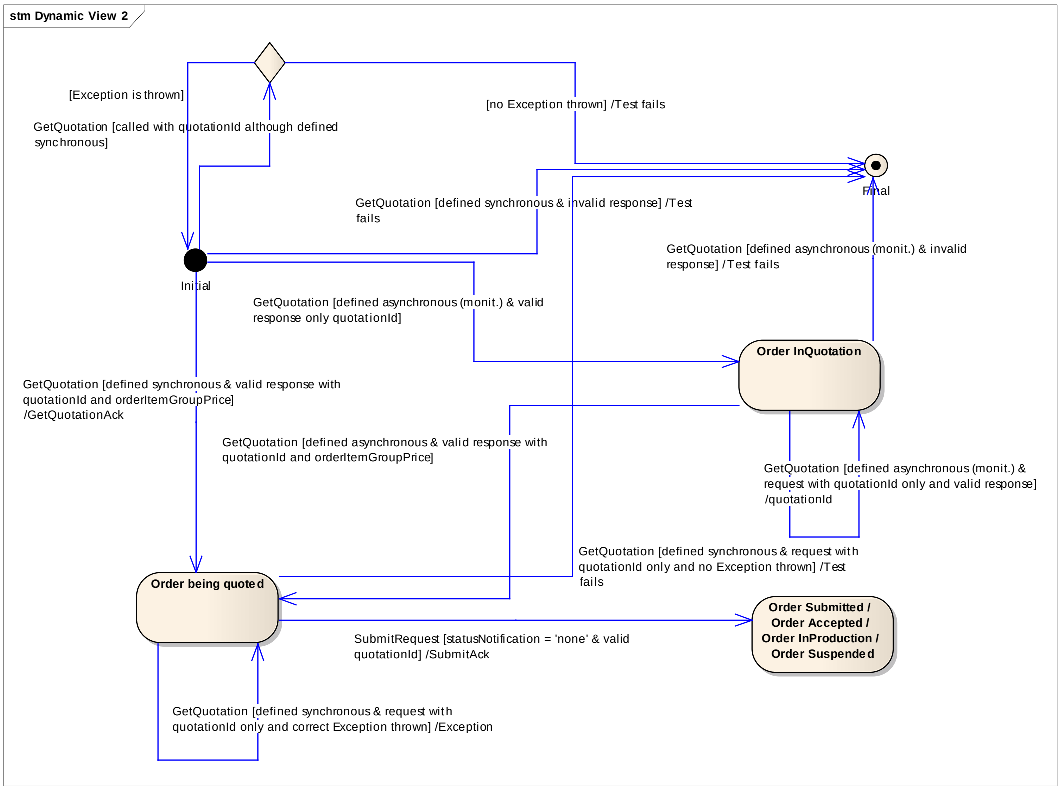
stm Dynamic View 2
[Exception is thrown]
GetQuotation [called with quotationId although defined synchronous]
[no Exception thrown] /Test fails
Final
GetQuotation [defined synchronous & invalid response] /Test fails
GetQuotation [defined asynchronous (monit.) & invalid response] /Test fails
Initial
GetQuotation [defined asynchronous (monit.) & valid response only quotationId]
Order InQuotation
GetQuotation [defined synchronous & valid response with quotationId and orderItemGroupPrice] /GetQuotationAck
GetQuotation [defined asynchronous & valid response with quotationId and orderItemGroupPrice]
GetQuotation [defined asynchronous (monit.) & request with quotationId only and valid response] /quotationId
GetQuotation [defined synchronous & request with quotationId only and no Exception thrown] /Test fails
Order being quoted
Order Submitted / Order Accepted / Order InProduction / Order Suspended
SubmitRequest [statusNotification = 'none' & valid quotationId] /SubmitAck
GetQuotation [defined synchronous & request with quotationId only and correct Exception thrown] /Exception

*Figure 8: EO Order State Diagram (synchronous GetQuotation operation)*

The configuration file of an executable test environment should define the Product Identifier and the Collection Identifier (where the Product Identifier belongs to – this is a predefined taskingRequestId in case of future product orders) which shall be ordered or subscribed to.

### 2.1.1 Conformance Class EOOrderCorePO

This conformance class defines test cases covering the support for the basic synchronous Order operations handling product orders.

### 2.1.2 Conformance Class EOOrderCoreSub

As EOOrderCorePO this conformance class defines test cases covering the support for the basic synchronous Order operations but handling subscriptions.

### 2.1.3 Conformance Class EOOrderCoreSPS

As EOOrderCorePO this conformance class defines test cases covering the support for the basic synchronous Order operations but handling tasking requests.

### 2.1.4 Conformance Class EOOrderCoreAsynch(PO|Sub|SPS)

These optional conformance classes (for PO/Sub/SPS) define test cases covering the optional additional support for asynchronous Submit requests. The Submit operation allows the client to specify a Callback handler.

### 2.1.5 Conformance Class EOOrderResultAccess

This optional conformance class defines test cases covering the support for the DescribeResultAccess operation.

### 2.1.6 Conformance Classes EOOrderQuotation(PO|Sub|SPS)

These conformance classes define test cases covering the GetQuotation support (orderTypes: PO, Sub or SPS) for the two different processing types (for the three product types):

- Synchronous quotation
- Asynchronous quotation with monitoring

### 2.1.7 Conformance Classes EOOrderQuotationAsynch(PO|Sub|SPS)

These conformance classes define test cases covering the optional additional support for asynchronous (with notification) GetQuotation requests. The GetQuotation operation allows the client to specify a Callback handler.

### 2.1.8 Conformance Class EOOrderCancel

This conformance class defines test cases covering the support for the synchronous Cancel operation.

### 2.1.9 Conformance Class EOOrderCancelAsynch

This optional conformance class defines test cases covering the optional additional support for asynchronous Cancel requests. The Cancel operation allows the client to specify a Callback handler.

## 2.2 Conformance Test Classes

### 2.2.1 Conformance Test Classes: EOOrderCore(PO|Sub|SPS)

This chapter defines generically three conformance test classes: EOOrderCorePO, EOOrderCoreSub and EOOrderCoreSPS. These test classes cover the testing of the basic synchronous Order operations in combination with one of the three order types: **PO** (for **product orders), Sub** (for **Subscriptions**) and **SPS** (for **future products** or **tasking requests**). Within the ETS a separate executable test class has to be defined for each of the three abstract test classes.

#### *2.2.1.1 GetCapabilities support*

a) Test purpose: The service under test (SUT[3]) correctly supports the HMA Order GetCapabilities operation.

b) Test method: Verify that the SUT accepts a valid HMA Order GetCapabilities request send via HTTP/KVP/GET. The response to the request should be valid to the requirements defined in Subclause 8.1.2.2. This includes a conformant description of the 'OperationsMetadata' section (defining all implemented operations) as well as a conformant description of the 'contents' section. Pass if the assertion is satisfied; fail otherwise.

c) References: OGC 06-141r2 Subclause 8.1.2, Table 8.1, C1.

d) Test type: Capability

#### *2.2.1.2 GetOptions support*

a) Test purpose: The service under test (SUT[4]) correctly supports the HMA Order GetOptions operation for an EO product.

b) Test method: Verify that the SUT accepts a valid HMA Order GetOptions request send via HTTP/XML/POST/SOAP 1.2. Depending on the order type under test the following ID has to be provided within the request:

[3] The abbreviation SUT will be used in the following for "service under test".

[4] The abbreviation SUT will be used in the following for "service under test".

I. The predefined collectionId in case of a product order (PO). If an exception is thrown a subsequent GetOptions call additionally providing the productId as request parameter will be executed as sometimes the options can only be acquired for the product (and not only for the collection). If then an exception is thrown again, the test will fail, otherwise the test will pass.

II. The predefined collectionId in case of a subscription (Sub).

III. The predefined taskingRequestId (SPS:ID) in case of future product order. The SPS:ID can refer different type of requests:

**Formattato:** Nessuna sottolineatura, Inglese (U.S.A.)

i. Precisely identifier acquisitions

ii. Coverage request

**Formattato:** Nessuna sottolineatura, Inglese (U.S.A.)

iii. Standing request

**Formattato:** Nessuna sottolineatura, Inglese (U.S.A.)

The response to the request should provide valid order options structured in (an) array(s) of groups, each identified by the productOrderOptionsId and representing a valid combination of options. If the final response is not valid, the test will fail, otherwise the test will pass.

c) References: OGC 06-141r2 Subclause 8.1.3, Table 8.1, C2.

d) Test type: Capability

#### *2.2.1.3 GetOptions error handling*

a) Test purpose: Verification that the SUT throws a correct error message on invalid (unknown) collectionId or invalid taskingRequestId in GetOptions request.

b) Test method: Send a valid HMA Order GetOptions request via HTTP/XML/POST/SOAP 1.2 with an unknown CollectionId (or unknown taskingRequestId) and receive in return an appropriate SOAP fault.

e) References: OGC 06-141r2 Subclause 8.1.3, Table 8.1, C2.

f) Test type: Capability

#### *2.2.1.4 Submit support: invalid productId*

a) Test purpose: Verification that the SUT throws a correct error message on submitting an invalid productId.

b) Test method: Send a valid HMA Order Submit request via HTTP/XML/POST/SOAP 1.2 for the predefined EO Product (order specification) specifying all the order parameters but a wrong productId. The response to the request should present an appropriate Submit acknowledge where “status” is set to “failure” and “orderId” is set “FAILURE”.

c) References: OGC 06-141r2 Subclause 8.1.6, C5.

d) Test type: Capability

#### *2.2.1.5 Submit support: invalid structured request*

e) Test purpose: Verification that the SUT throws a correct error message on invalid structured request.

f) Test method: Send an invalid structured (some tags are not compliant with the schema) HMA Order Submit request via HTTP/XML/POST/SOAP 1.2. The response to the request should present an appropriate Submit acknowledge where “status” is set to “failure” and “orderId” is set to “FAILURE”.

g) References: OGC 06-141r2 Subclause 8.1.6, C5.

h) Test type: Capability

#### *2.2.1.6 Submit support: incorrect order options request*

i) Test purpose: Verification that the SUT throws a correct error message on providing invalid order options within the request.

j) Test method: Send an HMA Order Submit request an request with invalid order options via HTTP/XML/POST/SOAP 1.2. The response to the request should present an appropriate Submit acknowledge where “status” is set to “failure” and “orderId” is set FAILURE”.

k) References: OGC 06-141r2 Subclause 8.1.6, C5.

l) Test type: Capability

#### *2.2.1.7 Submit support: invalid notification status*

a) Test purpose: Verification that the SUT throws a correct error message on invalid notification status.

b) Test method: Send a valid HMA Order Submit request send via HTTP/XML/POST/SOAP 1.2 for the predefined EO Product (order specification) specifying all the order parameters. The “statusNotification” parameter is set to “Final” (which is wrong for an Order Server which does not support Call-backs). The response to the request should present an appropriate SOAP fault.

c) References: OGC 06-141r2 Subclause 8.1.6, C5.

d) Test type: Capability

#### 2.2.1.8 Submit support

a) Test purpose: The service under test (SUT) correctly supports the HMA Order Submit operation.

b) Test method: Verify that the SUT accepts a valid HMA Order Submit request send via HTTP/XML/POST/SOAP 1.2 for the predefined EO Product (order specification) specifying all the order parameters (except scene Selection). “statusNotification” must be set to “none”. The response to the request should provide a valid submit acknowledge with “status” set to “success” and the “orderId” set.

c) References: OGC 06-141r2 Subclause 8.1.6, C5.

d) Test type: Capability

#### 2.2.1.9 Submit with SceneSelection Support

a) Test purpose: The service under test (SUT) correctly supports the HMA Order Submit operation with additional SceneSelection Options for an EO product order.

b) Test method: Verify that the SUT accepts a valid HMA Order Submit request (includeing sceneSelection Options) send via HTTP/XML/POST/SOAP 1.2 for the predefined EO Product (order specification) specifying all the order parameters. “statusNotification” must be set to “none”. The response to the request should provide a valid submit acknowledge with “status” set to “success” and the “orderId” set.

c) References: OGC 06-141r2 Subclause 8.1.6, C5.

d) Test type: Capability

#### 2.2.1.10 GetStatus Order Search brief support

a) Test purpose: Verification that the SUT correctly supports the HMA Order GetStatus operation (order search mode) for an EO product order.

b) Test method: Verify that the SUT accepts a valid HMA GetStatus order search request (brief response) send after the valid Submit request (see test case before) via HTTP/XML/POST/SOAP 1.2 specifying correctly all the GetStatus parameters. The response to this request should provide a valid brief OrderMonitor response. If no entry is returned on a filter on orderStatusInfo set to "Completed" the test should be repeated (after a short break). When within a predefined amount of time no entriy is returned the test should fail and stop testing this Conformance Class, otherwise it should pass and proceed with the next step.

c) References: OGC 06-141r2 Subclause 8.1.8, C8.

d) Test type: Capability

#### 2.2.1.11 GetStatus Order retrieve full support

e) Test purpose: Verification that the SUT correctly supports the HMA Order GetStatus operation (order retrieve mode) for an EO product order.

f) Test method: Verify that the SUT accepts a valid HMA GetStatus order retrieve request (full response) send after the valid Submit request and after the orderStatusInfo was already set to "Completed" (see test cases before) via HTTP/XML/POST/SOAP 1.2 specifying correctly all the GetStatus parameters. The response to this request should provide a valid full OrderMonitor response. If the orderStatusInfo within the response is not set to "Completed" the test should fail and stop testing this Conformance Class, otherwise it should pass and proceed with the next step.

g) References: OGC 06-141r2 Subclause 8.1.8, C8.

h) Test type: Capability

#### 2.2.1.12 GetStatus support: wrong OrderId

a) Test purpose: Verification that the SUT throws a correct error message on invalid orderId.

b) Test method: Verify that the SUT accepts an HMA GetStatus order retrieve request send via HTTP/XML/POST/SOAP 1.2 specifying correctly all the GetStatus parameters, except a wrong orderId, and receive in return an appropriate SOAP fault.

c) References: OGC 06-141r2 Subclause 8.1.8, C8.

d) Test type: Capability

#### 2.2.1.13 GetStatus support: invalid structured request

e) Test purpose: Verification that the SUT throws a correct error message on invalid structered request.

f) Test method: Verify that the SUT throws an appropriate SOAP fault on an invalid structured HMA GetStatus order retrieve request send via HTTP/XML/POST/SOAP 1.2.

g) References: OGC 06-141r2 Subclause 8.1.8, C8.

h) Test type: Capability

### 2.2.2 Conformance Test Classes: EOOrderCoreAsynch(PO|Sub|SPS)

This sub-chapter defines generically three additional and optional conformance classes: EOOrderCoreAsynchPO, EOOrderCoreAsynchSub and EOOrderCoreAsynchSPS. As defined by the test-cases 2.2.1.1 - 2.2.1.13 these conformance test classes cover the testing of the basic order operations (with one of the three order types PO, Sub or SPS) - but in this case with asynchronous (instead of synchronous) support of the “Submit” operation. Therefore these conformance test classes require the same test cases but 2.2.1.7-2.2.2.9 replaced by the following test case.

#### 2.2.2.1 Asynch Submit support

e) Test purpose: The service under test (SUT) correctly supports the asynchronous HMA Order Submit operation.

f) Test method: Verify that the SUT accepts a valid HMA Order Submit request send via HTTP/XML/POST/SOAP 1.2 for the predefined EO Product (order specification) specifying all the order parameters (except scene Selection). In this case the “statusNotification” must be set to “final”. The response to the request should provide a valid submit acknowledge with “status” set to “success” and the “orderId” set. After this the test client hat to wait for corresponding SubmitResponse call. If the SubmitResponse input message is correct the test will pass and proceed with the next steps otherwise fail.

g) References: OGC 06-141r2 Subclause 8.1.6, C5.

h) Test type: Capability

### 2.2.3 Conformance Test Class: EOOrderResultAccess

This Conformance Test Class is an optional “add-On” to the EOOrderCore(PO|Sub|SPS)-Test Classes. It will only be proceeded after a successful proceeding of the EOOrder-Core(PO|Sub|SPS)-Test Classes and if its procedure is configured.

#### *2.2.3.1 Online delivery support*

i) Test purpose: Verifies that the SUT correctly supports the HMA Order DescribeResultAccess operation for an EO product order, subscription or tasking request.

j) Test method: Verify that the SUT accepts a valid HMA DescribeResultAccess request send via HTTP/XML/POST/SOAP 1.2 specifying correctly all the DescribeResultAccess parameters. The “subFunction” parameter should be set to “all-ready”. The response to this request should provide a valid address of an “item” to be retrieved. If the “status” within the response is set to “success” the test passes, if the status is set to “partial” or “failure” the system should check if there is a correct “errorMessage”. If yes, the test passes as well. In all other cases the test will fail.

k) References: OGC 06-141r2 Subclause 8.1.9, C9.

l) Test type: Capability

### 2.2.4 Conformance Test Class: EOOrderCancel

This conformance class defines test cases covering the additional support of the cancel operation. This Conformance Test Class is an optional “slide-in” of the EOOrder-Core(PO|Sub|SPS)-Test Classes. It will be proceeded just after the submit-testcases and will replace the rest of the EOOrderCore(PO|Sub|SPS)-Test Classes (because the getStatus-test-cases have to be changed, this test class will provide new test cases for getStatus)

#### *2.2.4.1 Cancel support*

b) Test purpose: Verifies that the SUT correctly supports the HMA Order Cancel operation for an EO product order, subscription or tasking order.

c) Test method: Verify that the SUT accepts a valid HMA Order Cancel request send via HTTP/XML/POST/SOAP 1.2 specifying correctly all the Cancel parameters. ). In this case the “statusNotification” must be set to “final”. If the response to this request is correct (including the provision of a “status” element set to “success” or “incomplete”), the test should proceed. In all other cases the test will fail. After this the test client hat to wait for corresponding CancelResponse call. If the CancelResponse input message is correct the test will pass and proceed with the next steps otherwise fail.

d) References: OGC 06-141r2 Subclause 8.1.10, C10.

e) Test type: Capability

### 2.2.5 Conformance Test Class: EOOrderCancelAsynch

This sub-chapter defines an additional and optional conformance test class: EOOrderCancelAsynch. As defined by the test-case 2.2.4 this conformance test classes cover the testing of the cancel operation - but in this case with asynchronous (instead of synchronous) support. Therefore this conformance test class replaces 2.2.4.1 by the following test case.

#### *2.2.5.1 Asynch Cancel support*

f) Test purpose: Verifies that the SUT correctly supports the HMA Order Cancel operation for an EO product order, subscription or tasking order.

g) Test method: Verify that the SUT accepts a valid HMA Order Cancel request send via HTTP/XML/POST/SOAP 1.2 specifying correctly all the Cancel parameters. In this case the “statusNotification” must be set to “final”. The response to the request should provide a valid cancel acknowledge with “status” set to “success”. After this the test client hat to wait for corresponding SubmitResponse call. If the SubmitResponse input message is correct the test will pass and proceed with the next steps otherwise fail

h) References: OGC 06-141r2 Subclause 8.1.10, C10.

i) Test type: Capability

### 2.2.6 Conformance Test Class: EOOrderQuotation

This conformance class defines test cases covering the additional support of the GetQuotation operation. This Conformance Test Class is an optional “slide-in” of the EOOrderCore(PO|Sub|SPS)-Test Classes. It will be proceeded just after the GetOptions- and before the Submit-operation call. The Submit-operation call has to be adjusted, handling a “quotationId” request parameter instead of an “orderSpecification” request parameter.

The Test Class only checks two different GetQuotation related processing types:

- Synchronous quotation
- Asynchronous quotation with monitoring

Depending on the settings within the Capabilities document (XPath: Capabilities/Content/GetQuotationCapabilities) and depending on the content of the wsa:ReplyTo address within the GetQuotation request message the first or the second processing method will be tested.

#### *2.2.6.1 GetQuotation (synchronous) support – mistakenly called with "quotationId"*

a) Test purpose: Verifies that the SUT correctly supports the error handling for a synchronous HMA Order GetQuotation operation which is called mistakenly with a "quotationId" instead of an "orderSpecification".

b) Test method: Verify that the SUT accepts a HMA Order GetQuotation request send via HTTP/XML/POST/SOAP 1.2 specifying mistakenly a "quotationId" instead an "orderSpecification" substructure as input parameter. In this case a correct ows:Exception report must be thrown (otherwise the test fails).

c) References: OGC 06-141r2 Subclause 8.1.4, C3.

d) Test type: Capability

#### *2.2.6.2 GetQuotation (synchronous) – invalid structured request*

e) Test purpose: Verifies that the SUT correctly supports correct error handling on an invalid synchronous HMA Order GetQuotation request.

f) Test method: Verify that the SUT throws a correct ows:Exception report (otherwise the test fails) on an invalid structured HMA Order GetQuotation request send via HTTP/XML/POST/SOAP 1.2.

g) References: OGC 06-141r2 Subclause 8.1.4, C3.

h) Test type: Capability

#### *2.2.6.3 GetQuotation (synchronous) support*

i) Test purpose: Verifies that the SUT correctly supports the synchronous HMA Order GetQuotation operation for an EO product order, subscription or tasking request.

j) Test method: Verify that the SUT accepts a valid HMA Order GetQuotation request send via HTTP/XML/POST/SOAP 1.2 specifying correctly all the GetQuotation parameters, especially the "orderSpecification" substructure parameters. The "status" within the response must be set to "success" or "partial" (otherwise the test fails). Further the "quotation" parameter (specifying the quotation of the order) has to be filled correctly. This means that the quotation element must include a valid quotationId and minimally one orderItemGroupPrice element filled with the mandatory values. Otherwise the test fails The quotationId will in the next step be passed to the Submit request.

k) References: OGC 06-141r2 Subclause 8.1.4, C3.

l) Test type: Capability

#### *2.2.6.4 GetQuotation (asynchronous with monitoring) support*

a) Test purpose: Verifies that the SUT correctly supports the asynchronous (with monitoring) HMA Order GetQuotation operation for an EO product order, subscription or tasking request.

b) Test method: Verify that the SUT accepts a valid HMA Order GetQuotation request send via HTTP/XML/POST/SOAP 1.2 filling correctly at minimum all the mandatory GetQuotation request parameter values, especially the parameter values within the "orderSpecification" substructure. Within the response the value of the "status" element must be set to "success" or "partial" (otherwise the test will fail). Further in the response either the "quotation" structure (specifying the quotation of the order) has to be filled correctly or (alternativerly) the "quotationId" element has to be returned. A correctly filled "quotation" structure includes a valid "quotationId" and minimally one valid filled "orderItemGroupPrice" substructure. If a "quotation" element is returned but the content of this structure is not valid filled, the test will fail. The "quotationId" parameter value from within quotation element will (in the next step) be passed to the Submit request. If the response did not return a "quotation" element but only a "quotationId" the test system should first wait for a short time and then call the GetQuotation operation again, providing this "quotationId" as input parameter. This step is repeated as far as a valid "quotation" is returned (and the test will pass) or as far a as a predefined amount of time ran out or as an incorrect structure is returned and the test will fail.

c) References: OGC 06-141r2 Subclause 8.1.4, C3.

d) Test type: Capability

### 2.2.7 Conformance Test Classes: EOOrderGetQuotationAsynch(PO|Sub|SPS)

This sub-chapter defines generically three additional and optional conformance test classes: EOOrderGetQuotationAsynchPO, EOOrderGetQuotationAsynchSub and EOOrderGetQuotationAsynchSPS. As defined by the test-cases 2.2.6.1-2.2.6.4 these conformance test classes cover the testing of the GetQuotation operation (with one of the three order types PO, Sub or SPS) - but in this case asynchronous with notification (instead of synchronous or asynchronous with monitoring). Therefore these conformance test classes require the same test cases but 2.2.6.1.2.2.6.2 replaced by the following test case.

#### *2.2.7.1 GetQuotation (asynchronous with notification) support*

e) Test purpose: Verifies that the SUT correctly supports the asynchronous (with notification) HMA Order GetQuotation operation for an EO product order, subscription or tasking request.

f) Test method: Verify that the SUT accepts a valid HMA Order GetQuotation request send via HTTP/XML/POST/SOAP 1.2 filling correctly at minimum all the mandatory GetQuotation request parameter values, especially the parameter values within the "orderSpecification” substructure and <wsa:ReplyTo> set correctly. Within the response the value of the “status” element must be set to “success” or “partial” (otherwise the test will fail). Further in the response the “quotation” structure shall not be filled but instead the “quotationId” element has to be returned. After this the test client hat to wait for corresponding GetQuotationResponse call. If the GetQuotationResponse input message is correct (including a correctly filled “quotation” structure with a valid “quotationId” and minimally one valid filled “orderItemGroupPrice” substructure) the test will pass and proceed with the next steps otherwise fail. The “quotationId” parameter value from within quotation element will (in the next step) be passed to the Submit request. If the

g) References: OGC 06-141r2 Subclause 8.1.4, C3.

h) Test type: Capability