Open Geospatial Consortium

Submission Date: 2016-02-18

Approval Date: 2016-05-08

Publication Date: 2017-09-13

External identifier of this OGC® document: http://www.opengis.net/doc/IS/3dps/1.0

Internal reference number of this OGC® document: 15-001r4

Version: 1.0

Category: OGC® Implementation Standard

Editors: Benjamin Hagedorn, Simon Thum, Thorsten Reitz, Voker Coors, Ralf Gutbell

OGC® 3D Portrayal Service 1.0

Document type: OGC® Standard

Document subtype:

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.

Table of Contents

i. Abstract

The 3D Portrayal Service Standard is a geospatial 3D content delivery implementation specification. It focuses on what is to be delivered in which manner to enable interoperable 3D portrayal.

It does not define or endorse particular content transmission formats, but specifies how geospatial 3D content is described, selected, and delivered. It does not prescribe how aforementioned content is to be organized and represented, but provides a framework to determine whether 3D content is interoperable at the content representation level. More details are available in Design of this standard.

ii. Keywords

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

ogcdoc, OGC document, 3D, portrayal, service, geospatial, specification

iii. Preface This document is based on two OGC discussion papers (WPVS: OGC 09-166r2, W3DS: OGC 09-104r3), which each detail a service interface to support 3D portrayal of geodata, but using two different mechanisms (image and scene-graph based). Due to considerable overlap, it was decided that, for standardization, their technical content should be merged as far as possible. This document represents the efforts by the 3D Portrayal Service SWG to merge the two proposals, retaining these two different mechanisms while providing a foundation for other potential mechanisms.

This document does not suggest any updates to the OGC Abstract Specification.

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.

iv. Submitting organizations

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

- Fraunhofer Gesellschaft
- Hasso Plattner Institute at the University of Potsdam
- Esri R&D Center Zürich

v. Submitters

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

Table 1. Editors

Name	Affiliation
Benjamin Hagedorn	Hasso Plattner Institute at the University of Potsdam
Simon Thum	Fraunhofer IGD
Thorsten Reitz	Esri R&D Center Zürich
Volker Coors	Fraunhofer IGD, HFT Stuttgart
Ralf Gutbell	Fraunhofer IGD

Table 2. Additional contributors

Name	Affiliation
Arne Schilling	virtualcitySYSTEMS
Dieter Hildebrandt	Hasso Plattner Institute at the University of Potsdam
Jan Klimke	Hasso Plattner Institute at the University of Potsdam
Jürgen Döllner	Hasso Plattner Institute at the University of Potsdam
Mike McCann	Monterey Bay Aquarium Research Institute
Tatjana Kutzner	Technical University of Munich
Thomas H. Kolbe	Technical University of Munich

Chapter 1. Scope

This OGC® document specifies a standard service interface for web-based 3D geodata portrayal supporting a) delivery of geometric 3D scene data and b) server-side 3D scene rendering. It is applicable to 3D geodata stores that want to target a range of portrayal clients, or to 3D geodata portrayal clients that want to portray geodata from a range of compatible sources.

This standard intends to enable semantic interoperability in geospatial 3D portrayal services. That is, it represents 3D geodata with potentially rich metadata and accompanying description of the technical requirements for portrayal thereof on a given client. It addresses use cases such as 3D portrayal of geodata and requesting additional information at the user's discretion. More details about possible interoperability scenarios may be obtained from the 3DPIE report [OGC 12-075].

This standard does not define or endorse a transmission format for scenes or images. It is therefore not sufficient to enable interoperability, but to determine automatically if interoperation is possible.

Chapter 2. Conformance

This OGC interface standard targets at 3DPS 1.0 implementations, i.e., 3D portrayal services and clients.

Requirements for 4 standardization target types are considered:

- Conformance class *core*, of http://www.opengis.net/spec/3DPS/1.0/conf-class/core, with a single pertaining requirements class, *core*, of http://www.opengis.net/spec/3DPS/1.0/req/core.
- Conformance class *scene*, of http://www.opengis.net/spec/3DPS/1.0/conf-class/scene, with a single pertaining requirements class, *scene*, of http://www.opengis.net/spec/3DPS/1.0/req/scene.
- Conformance class *view*, of http://www.opengis.net/spec/3DPS/1.0/conf-class/view, with a single pertaining requirements class, *view*, of http://www.opengis.net/spec/3DPS/1.0/req/view.
- Conformance class *info*, of http://www.opengis.net/spec/3DPS/1.0/conf-class/info, with a single pertaining requirements class, *info*, of http://www.opengis.net/spec/3DPS/1.0/req/info.

Conformance with this standard shall be checked using all the relevant tests specified in Annex A (normative) of this document. The framework, concepts, and methodology for testing, and the criteria to be achieved to claim conformance are specified in the OGC Compliance Testing Policies and Procedures and the OGC Compliance Testing web site.

In order to conform to this OGC® interface standard, a software implementation shall implement the *core* conformance class and one or more of the *scene* and *view* conformance classes. It is also possible for a yet unspecified extension to be considered sufficient for conformance as are the *scene* and *view* extensions. Such an extension shall be accepted as an addendum to this standard, facilitate 3D portrayal and provide at least one conformance class that is dependent on the *core* conformance class.

Requirements URIs and conformance test URIs defined in this document are relative to http://www.opengis.net/spec/3DPS/1.0.

It has been brought to the SWGs attention that the omission of a baseline format has consequences for the testability of the service specification and hence, the definition of conformance. Namely, in the case of the *scene* conformance class, no single delivery format may be used to define or test conformance. However this mirrors the situation in the 3D modeling world and seems unlikely to change soon. Thus, an abstract statement of conformance to the 3D portrayal service standard is to be seen as a first step to make it easier to interoperate. A particular service instance may not work with a given client due to differences in data encoding, format provisions employed, or a focus on the image-based or scene-based conformance classes. However, a client implementation will be able to tell reliably whether and how interoperation with a given service instance is possible.

Chapter 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.

OGC 04-046r3

Open Geospatial Consortium, OGC 04-046r3, *The OpenGIS Abstract Specification, Topic 2: Spatial Referencing by Coordinates*, August 2004

OGC CityGML

Open Geospatial Consortium, OGC OGC 12-019, *OGC City Geography Markup Language (CityGML) Encoding Standard*, Version: 2.0.0

OWS Common

Open Geospatial Consortium, OGC 06-121r9, *OGC Web Services Common Standard*, version 2.0

W3C XML 1.0

W3C Recommendation, *Extensible Markup Language (XML) 1.0 (Fifth Edition)*, http://www.w3.org/TR/xml

IETF RFC 1738

IETF RFC 1738, *Uniform Resource Locators (URL)*

IETF RFC 2046

N. Fred, N. Borenstein, *Multipurpose Internet Mail Extension (MIME) Part Two: Media Types*, November 1998

ISO/IEC 14977

ISO/IEC 14977:1996(E), *Extended BNF*

ISO 19117

ISO 19117:2012, *Geographic information — Portrayal*

RFC 3986

IETF RFC 3986, *Uniform Resource Identifier (URI): Generic Syntax*

RFC 2046

IETF RFC 2046, N. Freed, N. Borenstein, *Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types*, Nov. 1998

OGC KML

OGC 07-147r2, *OGC KML*, 2008

In addition to this document, this standard includes several normative XML Schema files. Following approval of this document, these schemas will be posted online at the http://schemas.opengis.net. These XML Schema files are also bundled with the present document. In the event of a discrepancy

between the bundled and online versions of the XML Schema files, the online files shall be considered authoritative.

Chapter 4. Terms and Definitions

For the purpose of this document, the terms and definitions given in the above references apply. In addition, the following terms and definitions apply.

Portrayal

presentation of information to humans.
NOTE: This term is defined by [ISO 19117].

Scene, 3D scene

geometry and texture data that is to be portrayed.

View, 3D view

rendering result generated by projecting a 3D scene to a view plane.

Chapter 5. Conventions

This section provides details of conventions used in the document.

5.1. Use of the terms "3D scene" and "3D view"

The term "3D scene" refers to a digital representation of geographic data that is mainly composed from 3D graphics data (mainly geometry and texture data), which is also often referred to as 3D display elements.

The term "3D view" refers to a visual representation of a 3D scene that was created by a 3D rendering system and can be directly perceived by humans.

5.2. Abbreviated terms

The following symbols and abbreviated are used in this standard.

3DPIE

3D Portrayal Interoperability Experiment (see [OGC 12-075])

3DPS

3D Portrayal Service

CS

Coordinate System

W3DS

Web 3D Service

WSC

OGC Web Services Common

WVS

Web View Service

5.3. UML notation

UML static structure diagrams appearing in this specification are used as described in Subclause 5.2 of [OWS Common].

5.4. Data dictionary tables

The UML model data dictionary is specified herein in a series of tables. The contents of the columns in these tables are described in Subclause 5.5 of [OWS Common]. The contents of these data dictionary tables are normative, including any table footnotes.

The "Names" column of these tables contains 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 Subclause 11.6.2 of [OWS Common].

For the reader's convenience, table rows describing inherited components are shaded.

5.5. Namespace prefix conventions

The following namespaces are used in this document (Table 3). The prefix abbreviations constitute conventions used here, but are not normative. The namespaces to which the prefixes refer are normative, however.

Table 3. Namespaces used in this document

Prefix	Namespace URI	Description
xsd	http://www.w3.org/2001/XMLSchema	XML Schema namespace
core	http://www.opengis.net/3dps/1.0/core	3DPS Version 1.0 Core
scene	http://www.opengis.net/3dps/1.0/scene	3DPS Version 1.0 Scene Extension
view	http://www.opengis.net/3dps/1.0/view	3DPS Version 1.0 View Extension
info	http://www.opengis.net/3dps/1.0/info	3DPS Version 1.0 Info Extension

Chapter 6. 3D Portrayal Service Overview

6.1. Overview

The 3D Portrayal Service (3DPS) is an OGC service implementation specification targeting the delivery of 3D portrayals in an interoperable fashion. When client and service(s) involved share a common set of capabilities, it becomes possible to view and analyze 3D geoinformation from diverse sources in a combined manner.

Major use cases include navigating in the represented scene, retrieving feature information, and analyzing detailed information like simulation results or other 3D spatial information provided using the service instances. See the OGC *Spatial Data on the Web Use Cases & Requirements* document [OGC 15-074 R1] for additional use cases for 3D on the Web.

6.2. Historical background

In the OGC Military Pilot Project, Phase 1, (MPP-1) three-dimensional portrayal was defined as a new operation — GetView — on a Web Map Service (WMS). As three-dimensional portrayal adds complexities that are out of scope of a WMS, a new Web Terrain Service (WTS) had been defined [OGC 01-061]. Later on, the different aspects and use cases of interoperable 3D geovisualization were discussed in detail by (Altmaier & Kolbe, 2003) within the context of the Geospatial Data Infrastructure North Rhine Westphalia (GDI NRW) in Germany. With the Web 3D Service (W3DS) they proposed a new web service delivering 3D computer graphics models. Two versions of the W3DS were published as OGC discussion papers [OGC 05-019] and [OGC 09-104r1]. OGC-internally, the development of 3D portrayal capabilities led to the proposal of a Web Perspective View Service (WPVS, serving rendered image data), which was generalized and published as OGC Web View Service (WVS) discussion paper [OGC 09-166r2].

Subsequently, five service implementations of at least one of both standards, together with 5 clients (both tailored and general-purpose), were subjected to the 3D Portrayal Interoperability Experiment (3DPIE, final report published as [OGC 12-075]). It emerged that several interoperability scenarios combining both approaches were indeed possible, and that the differences between W3DS and WVS were significant but mostly reconcilable.

However, some weaknesses also emerged. For example, the problem of scaling to bigger geodata was tackled with the well-known tiling technique. Tiling does not easily translate to geometric 3D data, and, as a consequence, there is no one-size-fits-all solution. Despite this, the proposals put forward a limited but complex solution.

The 3DPS combines the essential parts of the proposed W3DS and WVS into one common interface. It intentionally does not address some features, notably tiling, to the degree previous approaches did. However the 3D Portrayal Service SWG recognizes the potential of tiling and welcomes work on standardization to support tiling orthogonally to this standard.

In particular, any kind of approach that is based on spatial index transmission format (as could be negotiated and delivered through a 3DPS implementation) will likely work well in a 3DPS service implementation and benefit from the rich metadata provided through OGC Web Services Common

[OWS Common], the portable negotiation of format-specific idiosyncrasies, and the seamless transition to server-side rendering offered by 3DPS. Several such formats are standardized, e.g., [OGC KML]. At the time of this writing, multiple efforts to set industry standards that address the problem of tiling and indexing 3D geodata are also under way.

6.3. Design of this standard

For this first version of a 3D Portrayal Service standard, the goal was to find a unifying core of 3D portrayal tasks that can be standardized as [OWS Common] based requests in that they provide potential for interoperability, do not limit the possible implementations too much, and retain wiggle room for upcoming technologies. It is considered in-scope to define how instances of this specification may establish interoperability in their particular case, but it is considered out-of-scope to define a single baseline for interoperability.

For example, it was clear that new approaches that stream images from live 3D renderings as moving pictures or that provide 3D scenes as a stream of geometries could not be expected to be amenable to standardization efforts at the time.

At the same time, refinement of 3D scenes or 3D representations beyond approaches based on multiple representations were not in the base discussion papers and had to be left out due to a lack of agreement on the semantics of such approaches.

3D portrayal, in particular the scalable sort, poses specific challenges to the design of the underlying data storage and query capabilities, which are partly subject to this standard and partly have to be left to implementations, simply as there is no stable standardization target yet. Thus, it is expected that the number of actually interoperable implementations will lag behind adoption numbers, especially for the scene-based approach (as represented by the scene conformance class). Image-based portrayal, on the other hand, is easier to standardize due to well-established image formats.

This background is reflected in the standard by means of several design decisions. The standard

- defines two portrayal modes with a common core to stay adaptable to the moving target of 3D content representation and distribution technologies,
- re-uses existing format capabilities for delay-loading scene parts (tiling and streaming),
- defines many semantics as open lists of capabilities to provide flexible semantics,
- favors the possibility of determining interoperability of given implementations over decisions, which would actually enhance interoperability, to leave enough room for future innovation.

While these are relatively defensive design goals, the SWG believes that this set represents a good way to a useful first interface supporting interoperable service-based 3D portrayal. The goal of 3DPS is to enable actual interoperation, i.e., to automate matching 3D portrayal clients to services.

6.4. Interoperability scenarios

The interoperability scenarios that this standard enables mirror those demonstrated in the 3DPIE experiments (see OGC 12-075), namely [1: Experiment 1 is out of scope]:

- Linking W3DS and WVS (experiment 2),
- Integration of multiple W3DS in a 3D client (experiment 3),
- W3DS/WVS for browser-based portrayal (experiment 4),
- W3DS/WVS for mobile portrayal (experiment 5).

However, it is expected that more interoperability scenarios are enabled through this 3DPS standard.

Chapter 7. 3DPS Service Model

7.1. 3DPS operation types

The specified 3D Portrayal Service (3DPS) provides geometric 3D graphics data and/or rendered images. Thus, it supports two fundamental 3D portrayal schemes and associated client/server configurations.

The 3D Portrayal Service interface specifies the following operations that may be invoked by a 3DPS client and may be performed by a 3DPS service.

a. *GetCapabilities* — This operation allows a client to request information about a 3DPS server's capabilities and scene information offered.

b. *AbstractGetPortrayal* (abstract) — This is the abstract operation that forms the basis of the 3DPS operations *GetScene* and *GetView* and provides common parameters.

c. *GetResourceById* — This operation allows a client to request arbitrary resources, as indicated by the service.

d. *GetScene* — This operation allows a client to retrieve a 3D scene represented as 3D geometries and texture data, organized as a scene graph and/or spatial index.

e. *GetView* — This operation allows a client to retrieve a 3D view of a scene represented as images.

f. *AbstractGetFeatureInfo* (abstract) — This is the abstract operation that forms the basis for specific *GetFeatureInfo* operations that allow a client to retrieve more information about portrayed features.

g. *GetFeatureInfoByRay* — This operation allows a client to retrieve information about features that are selected based on a virtual ray.

h. *GetFeatureInfoByPosition* — This operation allows a client to retrieve information about features that are selected based on location.

i. *GetFeatureInfoByObjectId* — This operation allows a client to retrieve information about features that are selected based on object identifiers.

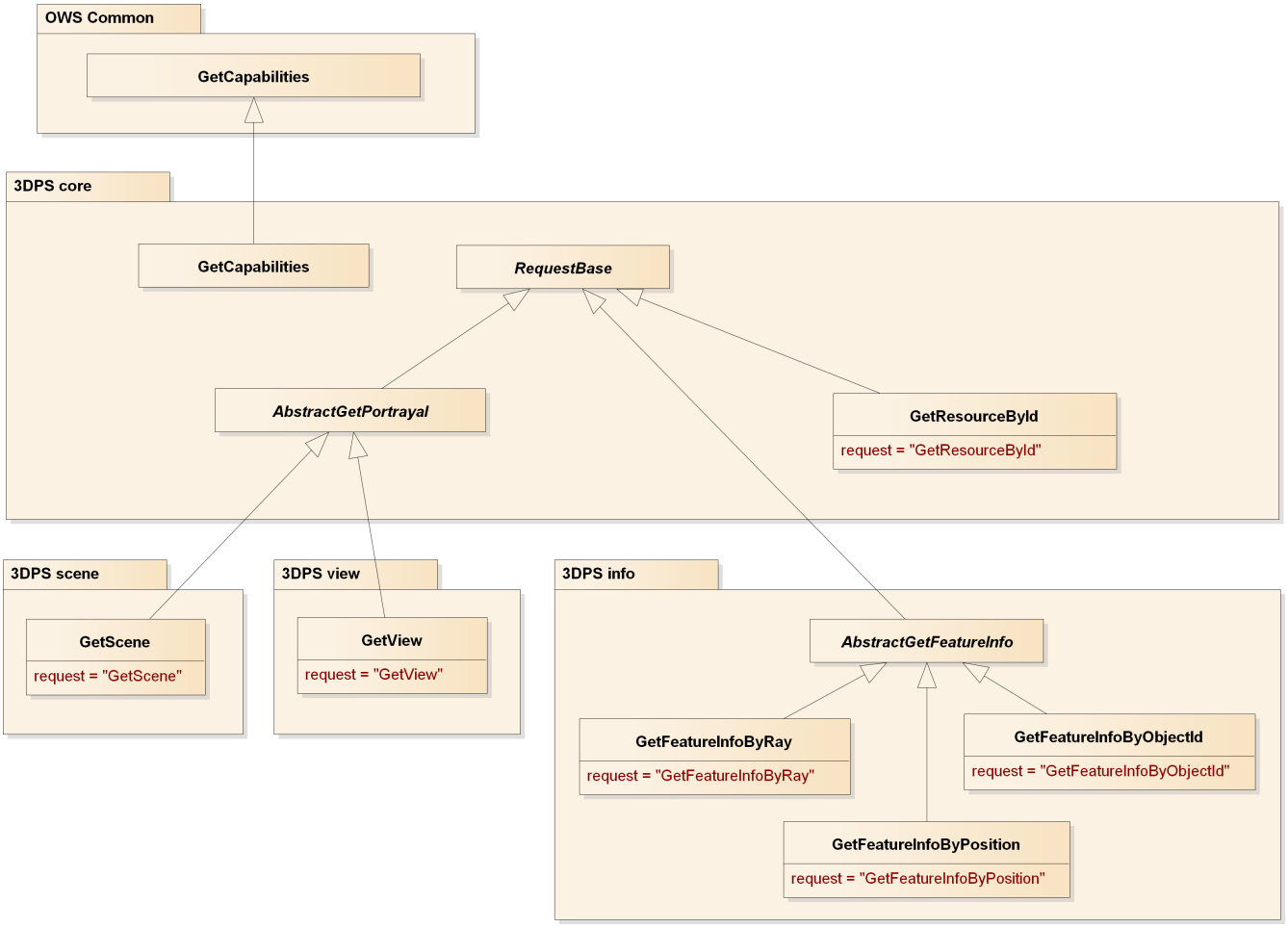

Figure 1. 3DPS UML class diagram

Figure 1 illustrates the 3DPS structure and interface as a UML class diagram. It shows that the 3DPS inherits the GetCapabilities operation from the [OWS Common] and adds the 3DPS operations.

A client should first, during a sequence of 3DPS requests, issue a GetCapabilities request to the service to obtain an up-to-date listing of supported operations and available data. To retrieve a vector representation or image representation of the data, a client will then perform one or more GetScene or GetView requests. If the client needs non-portrayal information (attributes) of one of the features, it will issue one of the GetFeatureInfo operations, depending on service capabilities and the information that is available to the client.

7.2. 3DPS service handling

The 3DPS operation requests except the GetCapabilities operation make use of the abstract core:RequestBase structure, which mimics the abstract RequestBase data structure from [OWS Common] Subclauses 9.2.

Accordingly, all 3DPS requests except GetCapabilities **shall** include, in addition to operation-specific parameters, the parameters described in Figure 2 and specified in Table 4:

- For all 3DPS requests, the request service parameter **shall** have a fixed value of ``3DPS''.
- For all 3DPS requests, the request version parameter **shall** have a fixed value of ``1.0''.
- The Extensions component is a hook for further request parameters defined, e.g., by 3DPS extension standards.

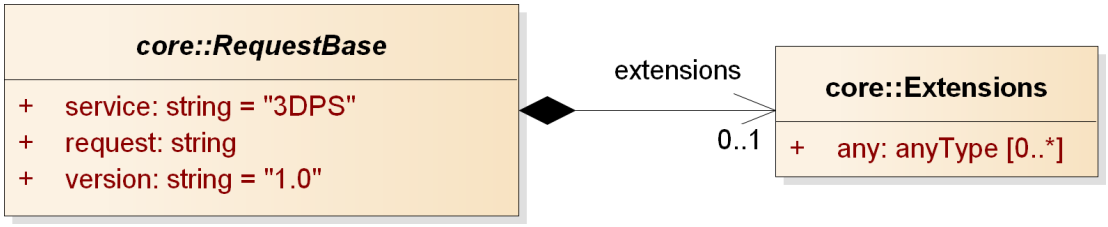

Figure 2. 3DPS core:RequestBase UML class diagram

Table 4. Components of core:RequestBase structure

Names	Definition	Data type and value	Multiplicity and use
service service	Service type identifier	string, not empty Value is fixed to ``3DPS''	One (mandatory)
request request	Operation name	string, not empty Value is operation name	One (mandatory)
version version	Standard version for operation	string, not empty Value is fixed to ``1.0''	One (mandatory)
extensions Extensions	Container for any kind of ancillary information to be sent from client to server	Extensions type	Zero or one (optional)

7.3. Coordinate systems

7.3.1. Coordinate reference systems

This standard uses several coordinate reference system (CRS) types, see Table 5. The two most important CRS types are the request CRS and the layer CRS. There might also be a bounding box CRS that is different from the request CRS.

Since CRS transformation and conversion is not necessarily feasible for some given client, it is recommended for an implementation to ensure a common CRS exists that is available on every layer.

Table 5. 3DPS Coordinate reference system types

Name	Definition
Request CRS	The CRS specified after the CRS parameter of a request. It is being considered for viewpoints and bounding boxes in request parameters and as the primary CRS in the response.
Bounding box CRS	A bounding box may be given with an explicit CRS specification. The bounding box CRS is either the explicitly specified CRS of the bounding box, or the request CRS.
Layer CRS	The layer CRS is (one of) the CRS in which a particular layer is represented. Depending on service capabilities, data from the layer may not be requested if the request CRS is not one of the layer CRSs.
Viewpoint CRS	The viewpoint CRS is the CRS in which a viewpoint is defined.

Vertical datum

In addition, the issue of a vertical datum is important for 3D portrayal because it defines the third dimension. At the time of this writing, there is no agreement on how to specify a vertical datum in service interfaces, so the vertical datum is implied in many cases. To enable communicating the vertical datum assumed by a service instance, each layer that holds height data **shall** advertise the vertical datum as a layer CRS. This CRS is then implied in requests querying the layer.

If no such CRS is specified, the vertical datum should be considered unknown, with undefined behavior potentially ensuing.

NOTE	This approach is subject to change as agreement is reached on handling vertical CRS in geospatial services.

7.3.2. Image coordinate system

An image CS is a coordinate system (CS) for a 3D view produced by a 3DPS supporting the *View* Extension. A 3D view is a rectangular grid of pixels. The image CS has a horizontal axis denoted x, and a vertical axis denoted y. x and y **shall** have only nonnegative integer values. The origin (x,y) = (0,0) is the pixel in the upper left corner of the image; x increases to the right and y increases downward. The image CS corresponds to the Map CS described in the WMS specification [OGC 06-042] clause 6.7.2.

The Width and Height parameters used in several 3DPS operation requests correspond to x and y as follows:

- Width denotes the size of the 3D view image in pixels along the x axis (that is, Width-1 is the maximum value of x).
- Height denotes the size of the 3D view image in pixels along the y axis (that is, Height-1 is the maximum value of y).

Chapter 8. 3DPS Core

8.1. Shared aspects

8.1.1. Position2D data structure

As defined in Figure 3 and Table 6, Position2D consists of two coordinates. If any of these coordinates is missing or empty, a 3DPS **shall** raise an InvalidParameterValue exception with the name of the parent element (in case of XML request) or the containing parameter (in case of HTTP/GET request).

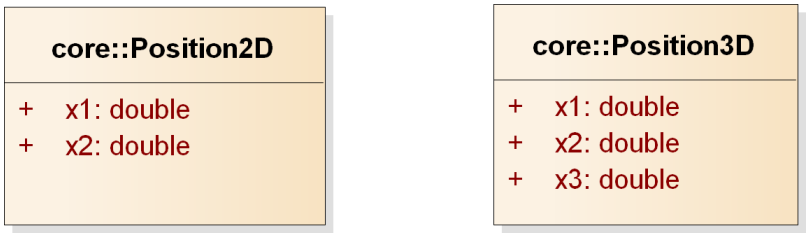

Figure 3. 3DPS core:Position2D and core:Position3D UML class diagram

Table 6. Components of core:Position2D structure

Names	Definition	Data type and value	Multiplicity and use
x1 X1	First position coordinate	Double Origin and units specified by request CRS	One (mandatory)
x2 X2	Second position coordinate	Double Origin and units specified by request CRS	One (mandatory)

8.1.2. Position3D data structure

As defined in Figure 3 and Table 7, Position3D consists of three coordinates. If any of these coordinates is missing or empty, a 3DPS service **shall** raise an InvalidParameterValue with the name of the parent element (in case of XML request) or the containing parameter (in case of HTTP/GET request).

Table 7. Components of core:Position3D structure

Names	Definition	Data type and value	Multiplicity and use
x1 X1	First position coordinate	Double Origin and units specified by request CRS	One (mandatory)
x2 X2	Second position coordinate	Double Origin and units specified by request CRS	One (mandatory)

Names	Definition	Data type and value	Multiplicity and use
x3 X3	Third position coordinate	Double Origin and units specified by request CRS	One (mandatory)

8.2. GetCapabilities operation (mandatory)

A GetCapabilities operation, as required by [OWS Common], allows a 3DPS client to retrieve service and scene metadata offered by a 3DPS server.

8.2.1. GetCapabilities request

Requirement 1: http://www.opengis.net/spec/3DPS/1.0/req/service/core/getcapabilities/request/structure

A core:GetCapabilities request **shall** consist of a core:GetCapabilities data structure that is derived from ows:GetCapabilities as specified in [OWS Common] Subclause 7.2.1 and that is refined as specified in Figure 4 and Table 8.

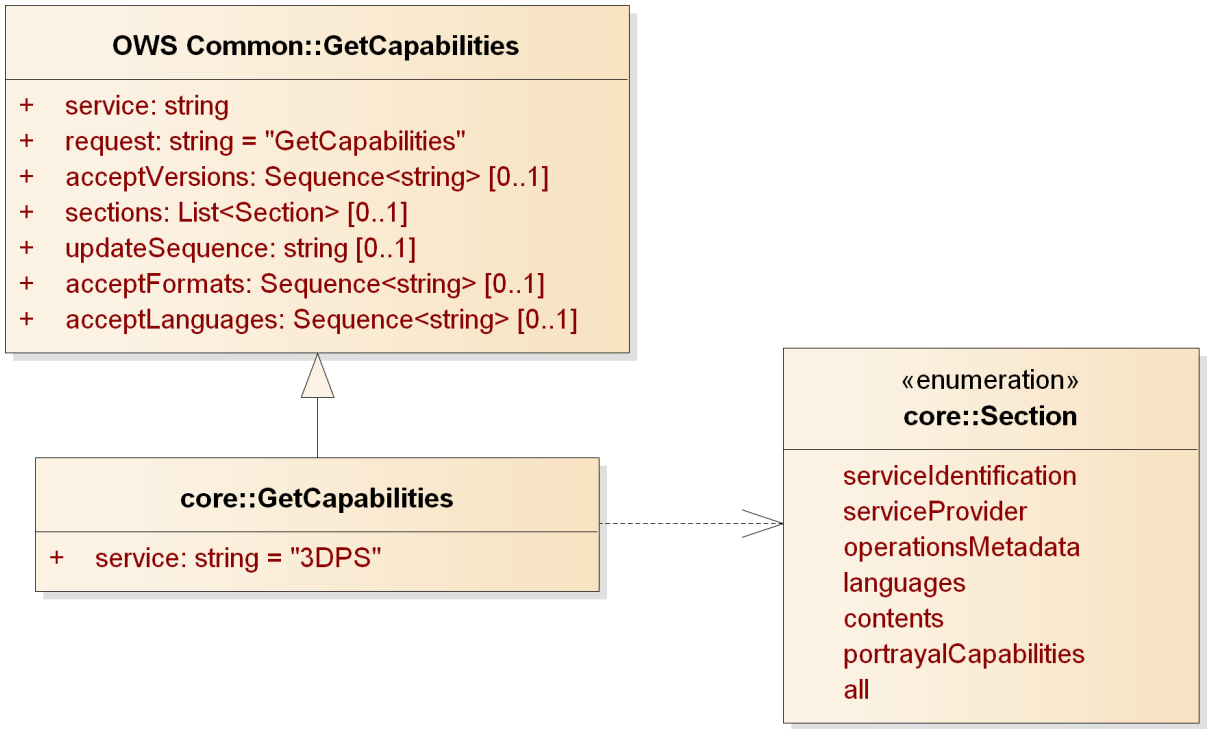

Figure 4. 3DPS core:GetCapabilities request UML class diagram

Table 8. Modified components of core:GetCapabilities request

Names	Definition	Data type and value	Multiplicity and use
service service	Service type identifier	string, not empty Value is fixed to ``3DPS''	One (mandatory)

Sections parameter

According to [OWS Common] Subclause 7.3.3., the Sections parameter value **shall** contain an

unordered list of zero or more names of the elements within a service metadata document that shall be returned. Table 9 lists the sections name values that are allowed; it includes the section name values specified by [OWS Common] and adds the PortrayalCapabilities Section.

Table 9. Meaning of allowed section name values

Section name	Meaning
Service Identification	Return ServiceIdentification element in service metadata document
ServiceProvider	Return ServiceProvider metadata element in service metadata document
OperationsMetadata	Return OperationsMetadata element in service metadata document
Languages	Return Languages metadata element in service metadata document
Contents	Return Contents metadata element in service metadata document
PortrayalCapabilities	Return PortrayalCapabilities metadata element in service metadata document
All	Return complete service metadata document, containing all elements

8.2.2. GetCapabilities response

A service metadata document **shall** be the normal response to a client performing a GetCapabilities request, and **shall** contain metadata appropriate to the specific 3DPS server. It includes metadata as defined in Subclause 7.4.2 of [OWS Common], a Contents section, and a PortrayalCapabilities section.

Requirement 2: http://www.opengis.net/spec/3DPS/1.0/req/service/core/getcapabilities/response/structure

The response to a successful GetCapabilities request **shall** consist of a core:Capabilities structure as defined in Figure 5, Table 10, Figure 6, Table 11, Table 12, Figure 7, and Table 14.

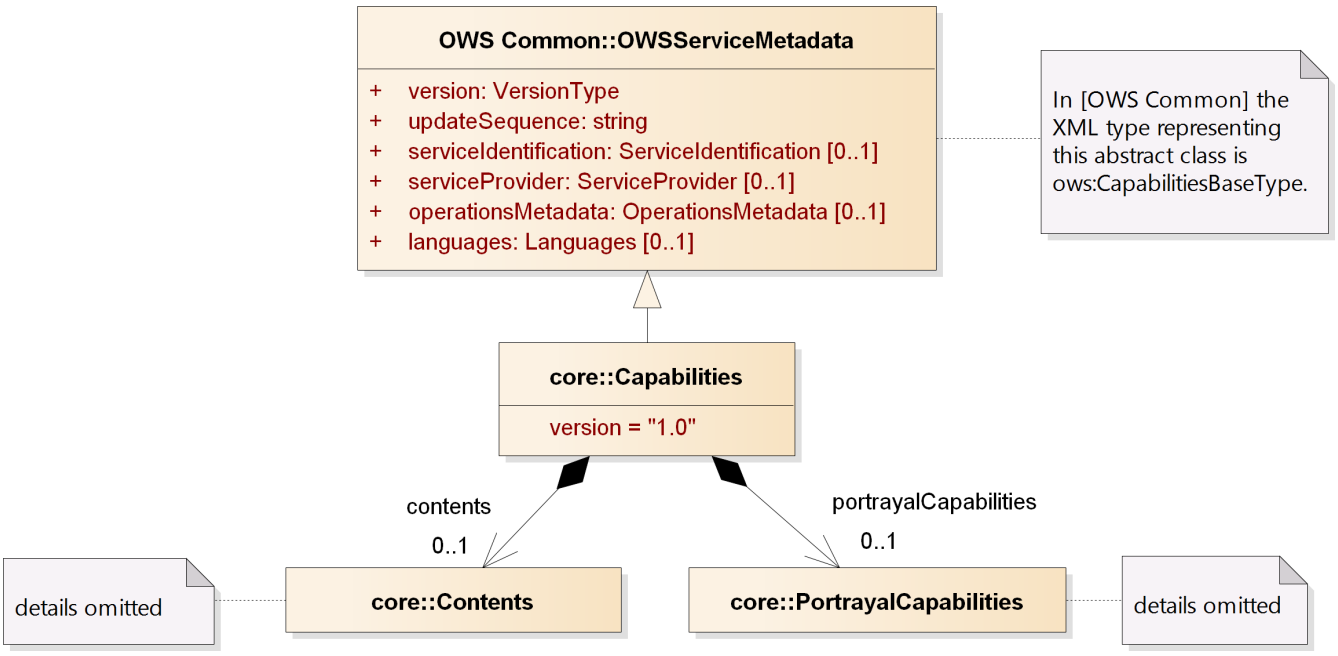

Figure 5. 3DPS core:Capabilities UML class diagram

Table 10. Components of core:Capabilities structure

Names	Definition	Data type and value	Multiplicity and use
version Version	Specification version for operation	string Value fixed to ``1.0''	One (mandatory)
updateSequence UpdateSequence	Service metadata document version, value is increased whenever any change is made in complete service metadata document, see [OWS Common]	string type, not empty	Zero or one (optional)
serviceIdentification ServiceIdentification	Metadata about this specific service. The schema of this section shall be the same as for all OWSs, as specified in Subclause 7.4.4 and owsServiceIdentification.xsd of [OWS Common].	ows:ServiceIdentification as defined in [OWS Common]	as defined in [OWS Common]
serviceProvider ServiceProvider	Metadata about the organization operating this service. The schema of this section shall be the same for all OWSs, as specified in Subclause 7.4.5 and owsServiceProvider.xsd of [OWS Common].	ows:ServiceProvider type as defined in [OWS Common]	as defined in [OWS Common]
operationsMetadata OperationsMetadata	Metadata about the operations specified by this service and implemented by this service, including the URLs for operation requests. The basic contents and organization of this section shall be the same as for all OWSs, as specified in Subclause 7.4.6 and owsOperationsMetadata.xsd of [OWS Common].	ows:OperationsMetadata type as defined in [OWS Common]	as defined in [OWS Common]
languages Languages	Languages supported by this server.	ows:Languages type as specified in [OWS Common], Subsection 7.4.9	as defined in [OWS Common]
contents Contents	Information about the 3D data content offered through this service	Contents type, see Table 11	Zero or one (optional)

Names	Definition	Data type and value	Multiplicity and use
portrayalCapabilities PortrayalCapabilities	Information about the portrayal capabilities of this service	PortrayalCapabilities type, see Table 14	Zero or one (optional)

Contents section contents

The Contents section of the service metadata document provides details about data layers that can be requested for portrayal. Its structure is derived from the OWSContents definition in [OWS Common], Subsection 7.4.8, as described in Figure 6, Table 11 and Table 12. The ows:DatasetSummary is replaced by core:Layer.

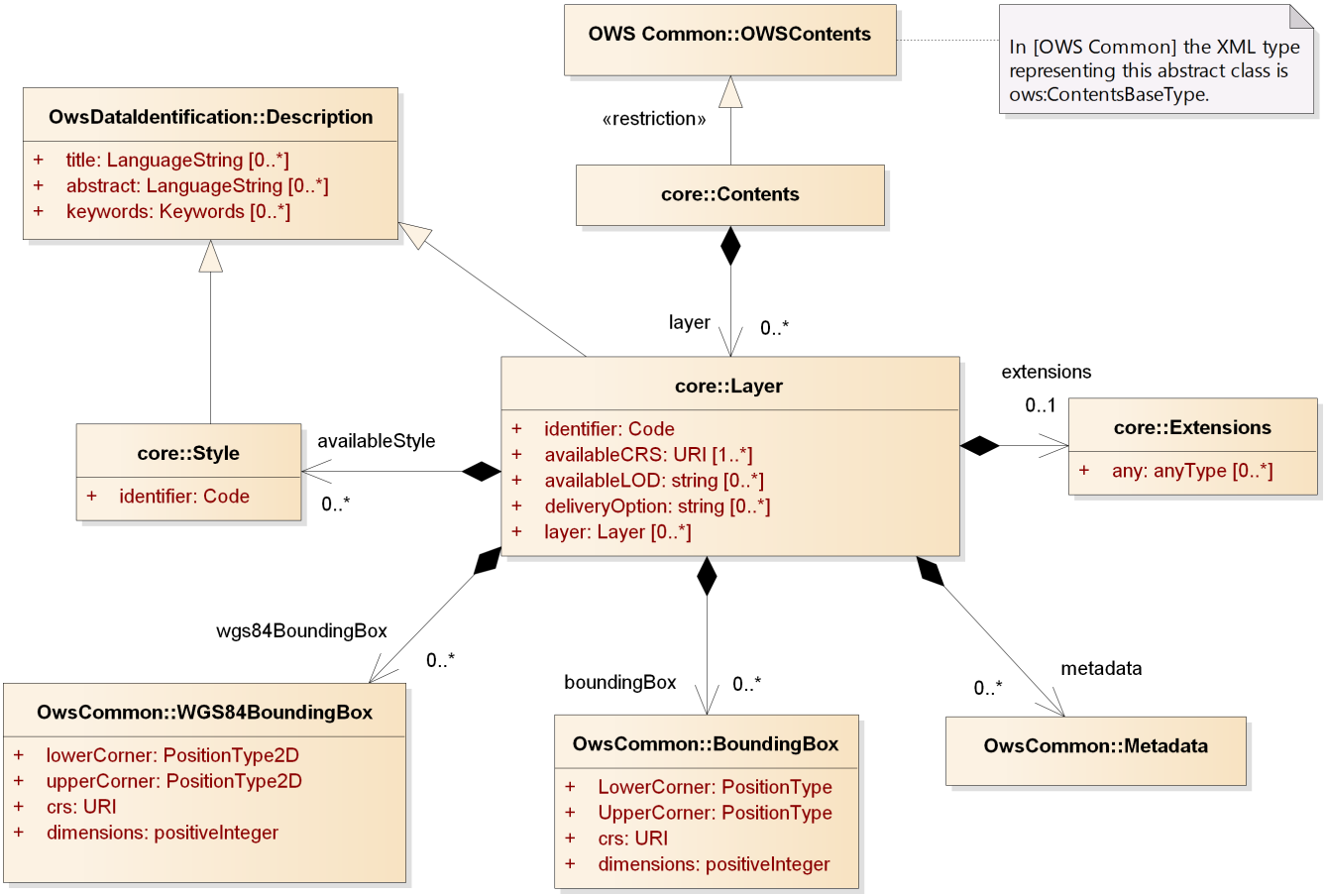

Figure 6. 3DPS core:Contents and core:Layer and UML class diagram

Table 11. Components of core:Contents structure

Names	Definition	Data type and value	Multiplicity and use
layer Layer	Metadata describing a dataset available from this service	Layer type, see Table 12	Zero or more (optional) Include as many as layers shall be advertised

Table 12. Components of core:Layer structure

Names	Definition	Data type and value	Multiplicity and use
title Title	Title of this dataset, human readable	ows:LanguageString, see [OWS Common] 10.7	Zero or more (optional)
abstract Abstract	Brief narrative description of this dataset	ows:LanguageString, see [OWS Common] 10.7	Zero or more (optional)
keywords Keywords	Unordered list of one or more commonly used or formalized word(s) or phrase(s) used to describe this dataset	ows:Keywords type in ows19115subset.xsd	Zero or more (optional) One for each keyword authority used
wgs84BoundingBox WGS84BoundingBox	Minimum bounding rectangle surrounding dataset, specified in WGS84 CRS with decimal degrees and longitude before latitude [a,c]	ows:WGS84BoundingBox, see [OWS Common], Table 34	Zero or more (optional) Include when useful or needed
identifier Identifier	Unambiguous identifier of this dataset, unique for this server	ows:Code type, not empty	One (mandatory)
boundingBox BoundingBox	Minimum bounding rectangle surrounding dataset, in available CRS [b,c]	ows:BoundingBox, see [OWS Common], Table 33	Zero or more (optional) Include when relevant and available, ideally at least one per AvailableCRS
availableCRS AvailableCRS	Coordinate reference system in which data from this layer may be requested	URI	One or more (mandatory)
availableLOD AvailableLOD	LOD value that holds data for this layer.	string	Zero or more (optional)
availableStyle AvailableStyle	Specification of style available for this layer	Style type, see Table 13	Zero or more (optional)
deliveryOption DeliveryOption	Identifier of delivery option available when requesting the layer.	string	Zero or more (optional) One for each supported delivery option
metadata Metadata	Reference to more metadata about this layer	ows:Metadata, see [OWS Common], Table 35	Zero or more (optional) Include when useful
extensions Extensions	Hook for layer extensions	Extensions type	Zero or one (optional)

Names	Definition	Data type and value	Multiplicity and use
layer Layer	Metadata describing one subsidiary dataset available from this service[d]	Layer type, see this table	Zero or more (optional) One for each subsidiary layer

[a] This WGS84BoundingBox can be approximate, but should be as precise as practical. If multiple WGS84 bounding boxes are included, this **shall** be interpreted as the union of the areas of these bounding boxes.
[b] More generally, definition of the horizontal, vertical, and temporal extent of this specific dataset. Zero or more BoundingBoxes are allowed in addition to one or more WGS84BoundingBoxes to allow more precise specification of the Dataset area in AvailableCRSs.
[c] If multiple bounding boxes are included having the same CRS, they **shall** be interpreted as their spatial union.
[d] Replaces the DatasetSummary component in ows:DatasetSummary

The components of the core:Layer are described and discussed in the following.

AvailableCRS

Every Layer is available in one or more layer coordinate reference systems. In order to indicate which layer CRSs are available, every named Layer **shall** have at least one AvailableCRS element that is either stated explicitly or inherited from a parent Layer. The root Layer **shall** include a sequence of zero or more AvailableCRS elements listing all CRSs that are common to all subsidiary layers. A child layer may optionally add to the list inherited from a parent layer. Any duplication **shall** be ignored by clients.

When a Layer is available in several coordinate reference systems, the list of available CRS values **shall** be represented as a sequence of AvailableCRS elements, each of which contains only a single CRS name.

EXAMPLE: <AvailableCRS>CRS:84</AvailableCRS> <AvailableCRS>EPSG:26718</AvailableCRS>.

AvailableLOD

A layer may contain objects in several representations to choose from. Thus, each layer may advertise a set of ``levels of detail" (LODs) present on that layer by referring to the LOD names as defined in the AvailableLODScheme elements of the PortrayalCapabilities section. See AvailableLODScheme for more information on LODs.

AvailableStyle

For each Layer, a server may advertise layer-specific styles (service styles), which modify the appearance of feature representations retrieved through the 3DPS AbstractGetPortrayal operation. For each layer-specific style, it provides an AvailableStyle element of type Style, which is listed in Table 13.

Each Style consists of an Identifier, a Title which may be presented to the user, an Abstract and a list of Keywords. The Abstract should give a brief narrative description of how the visualization is influenced by the style. The style identifier is used in an AbstractGetPortrayal request's Style parameter.

If more than one available style is advertised for one Layer, the server **shall** declare one of those AvailableStyle items as default by settings its IsDefault property to ``true''. If only a single style is available for one layer, that style does not need to be advertised by the server and is implicitly used as the layer's default style.

Table 13. Components of core:Style structure

Names	Definition	Data type and value	Multiplicity and use
title Title	Title of this style, human readable	ows:LanguageString, see [OWS Common] 10.7	Zero ore more (optional)
abstract Abstract	Brief narrative description of this style	ows:LanguageString, see [OWS Common] 10.7	Zero or more (optional)
keywords Keywords	Unordered list of one or more commonly used or formalized word(s) or phrase(s) used to describe this style	ows:Keywords type in ows19115subset.xsd	Zero or more (optional) One for each keyword authority used
identifier Identifier	Unambiguous identifier of this style, unique for this server	ows:Code type, not empty	One (mandatory)
isDefault IsDefault	This style is used when no style is specified for this layer in the request	Boolean type	Zero or one (optional) Default is ``false''

DeliveryOption

For a layer several delivery options may be available a client can choose from. Thus, each layer may advertise a set of delivery options present on that layer by referring to the delivery option names as defined in the DeliveryOption elements of the PortrayalCapabilities section; see DeliveryOption.

Extensions

The Extensions component is provided as a canonical place for extensions to define layer-specific metadata and can be used, e.g., by 3DPS extension modules as a hook for operation-specific layer extensions.

PortrayalCapabilities section contents

The PortrayalCapabilities structure is specified in Figure 7 and Table 14, Table 15, Table 16, Table 17, Table 18, Table 19.

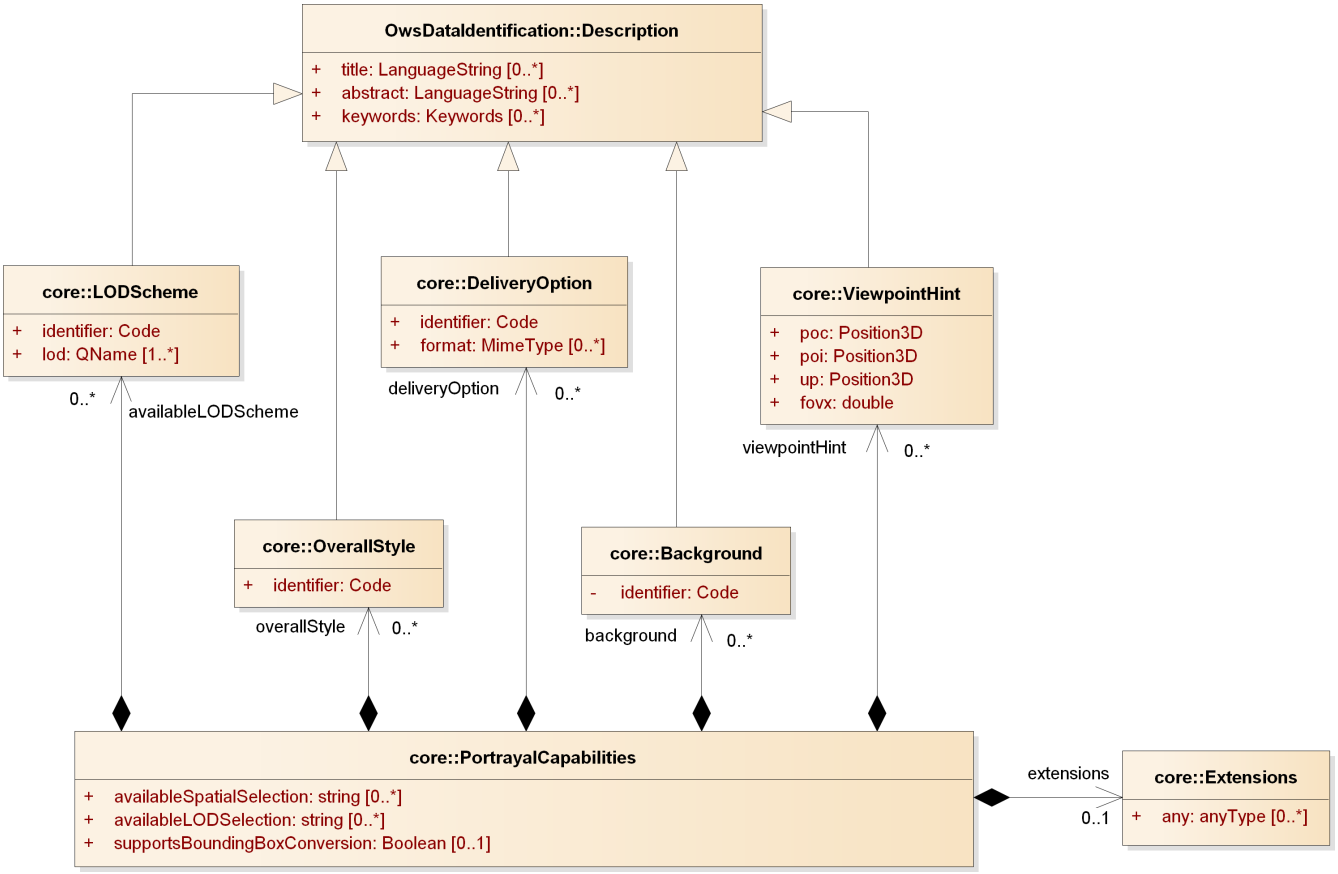

Figure 7. 3DPS core:PortrayalCapabilities UML class diagram

Table 14. Components of core:PortrayalCapabilities structure

Names	Definition	Data type and value	Multiplicity and use
overallStyle OverallStyle	Overall style supported	OverallStyle type, see Table 15	Zero or more (optional)
background Background	Metadata describing backgrounds that can be used for portrayal generation	Background type, see Table 18	Zero or more (optional)
availableSpatialSelection AvailableSpatialSelection	Spatial selection method supported	string, type+ Value is one of those in Table 20. Default is ``overlaps''	Zero or more (optional) Include when other than ``overlap'' method are offered

Names	Definition	Data type and value	Multiplicity and use
availableLODScheme AvailableLODScheme	LOD scheme supported	LODScheme type, see Table 16	Zero or more (optional)
availableLODSelection AvailableLODSelection	LOD selection method supported	string, Value is one of those in Table 22. Default is ``equals''	Zero or more (optional) Include when other than ``equals'' method is offered
deliveryOption DeliveryOption	Delivery option supported	DeliveryOption type, see Table 17	Zero or more (optional)
viewpointHint ViewpointHint	Metadata describing a meaningful viewpoint	ViewpointHint type, see Table 19	Zero or more (optional)
supportsBoundingBoxConversion SupportsBoundingBoxConversion	Flag indicating if the Server can convert non-advertised BoundingBoxes	Boolean type (true/false)	Zero or one (optional)
extensions Extensions	Hook for portrayal capability extensions	Extensions type	Zero or one (optional)

Table 15. Components of core:OverallStyle structure

Names	Definition	Data type and value	Multiplicity and use
title Title	Title of the overall style, normally used for display to a human	ows:LanguageString, see [OWS Common] 10.7	Zero or more (mandatory)
abstract Abstract	Brief narrative description of this overall style, normally available for display to a human	ows:LanguageString, see [OWS Common] 10.7	Zero or more (optional)

Names	Definition	Data type and value	Multiplicity and use
keywords Keywords	Unordered list of one or more commonly used or formalized word(s) or phrase(s) used to describe this overall style	ows:Keywords type in ows19115subset.xsd	Zero or more (optional) One for each keyword authority used
identifier Identifier	Unambiguous identifier of this overall style, unique for this server	ows:Code type, not empty	One (mandatory)

Table 16. Components of core:LODScheme structure

Names	Definition	Data type and value	Multiplicity and use
title Title	The name used to refer to the LOD scheme	ows:LanguageString, see [OWS Common] 10.7	Zero or more (optional)
abstract Abstract	Brief narrative description of this LOD scheme, normally available for display to a human	ows:LanguageString, see [OWS Common] 10.7	Zero or more (optional)
keywords Keywords	Unordered list of one or more commonly used or formalized word(s) or phrase(s) used to describe this LOD scheme	ows:Keywords type in ows19115subset.xsd	Zero or more (optional) One for each keyword authority used
identifier Identifier	Unambiguous identifier of this LOD scheme	ows:Code type, not empty	One (mandatory)
lod LOD	A level name as described in AvailableLODScheme	QName	One or more (mandatory)

Table 17. Components of core:DeliveryOption structure

Names	Definition	Data type and value	Multiplicity and use
title Title	Title of this delivery option, can be used for display to human	ows:LanguageString, see [OWS Common] 10.7	Zero or more (optional)
abstract Abstract	Brief narrative description of this delivery option, normally available for display to a human	ows:LanguageString, see [OWS Common] 10.7	Zero or more (optional)
keywords Keywords	Unordered list of one or more commonly used or formalized word(s) or phrase(s) used to describe this delivery option	ows:Keywords type in ows19115subset.xsd	Zero or more (optional) One for each keyword authority used

Names	Definition	Data type and value	Multiplicity and use
identifier Identifier	Unambiguous identifier or name of this delivery option, unique for this server	ows:Code type, not empty	One (mandatory)
format Format	A format which supports the delivery option. Empty means no restriction	ows:MimeType	Zero or more (optional)

Table 18. Components of core:Background structure

Names	Definition	Data type and value	Multiplicity and use
title Title	Title of the background, normally used for display to a human	ows:LanguageString, see [OWS Common] 10.7	Zero or more (optional)
abstract Abstract	Brief narrative description of this background, normally available for display to a human	ows:LanguageString, see [OWS Common] 10.7	Zero or more (optional)
keywords Keywords	Unordered list of one or more commonly used or formalized word(s) or phrase(s) used to describe this background	ows:Keywords type in ows19115subset.xsd	Zero or more (optional) One for each keyword authority used
identifier Identifier	Unambiguous identifier of this background, unique for this 3DPS server	ows:Code type, not empty	One (mandatory)

Table 19. Components of core:ViewpointHint structure

Names	Definition	Data type and value	Multiplicity and use
title Title	Title of this viewpoint, human readable	ows:LanguageString, see [OWS Common] 10.7	Zero or more (optional)
abstract Abstract	Brief narrative description of this viewpoint	ows:LanguageString, see [OWS Common] 10.7	Zero or more (optional)
keywords Keywords	Unordered list of one or more commonly used or formalized word(s) or phrase(s) used to describe this viewpoint	ows:Keywords type in ows19115subset.xsd	Zero or more (optional) One for each keyword authority used
poc POC	Position of the virtual camera (point of camera)	Position3D type, see Table 7	One (mandatory)
poi POI	Position of point of interest	Position3D type, see Table 7	One (mandatory)

Names	Definition	Data type and value	Multiplicity and use
up Up	Position of a point forming the vector pointing in ``up'' direction by subtracting the POC	Position3D type, see Table 7	Zero or one (optional) Include when camera roll desired

AvailableSpatialSelection

For selecting features to be part of a portrayal response, a server has to check the spatial relation between the request BoundingBox and the feature's 3D geometry.

In Table 20 the three possible spatial selection methods defined by this standard are named and described.

The default selection method is ``overlaps'', which includes any feature that is partly or entirely contained in the request BoundingBox into service response.

Table 20. Spatial selection methods

Name	Description
overlaps	A feature is only selected, if its 3D geometry is contained in or intersects with the given BoundingBox. This is the default mode.
contains_center	A feature is only selected, if its ``center point'' is contained or intersects with the BoundingBox. How the center point is computed by the service is not defined, but it **shall** be inside the convex hull of the feature.
cut	This spatial selection method **shall** not return features or parts of features that lie outside of the BoundingBox. Features that are completely contained in the BoundingBox shall be returned unmodified. Features that intersect with the borders of the BoundingBox shall be split and parts that lie outside shall be cut away. The parts that lie inside of the BoundingBox shall be selected for response. Multiple requests with adjacent BoundingBoxes shall generate feature geometries that fit seamlessly together without gaps or cracks.

OverallStyle

An OverallStyle is usually a reference to a scene or view embellishment that is not treated as being bound to a specific layer, e.g., an adornment of certain features, or a special shading to be used to portray a data overlay in the 3D scene or view. For simple clients, such embellishments may result in a much improved user experience, or show additional information. However, such offerings are often implementation-specific and may be harmful to service interoperability.

A server advertises available these portrayal-wide styles through a number of OverallStyle elements, each containing an OverallStyle name, which can be used by a client within a portrayal request.

Background

The Background parameter contains an available background as described in Table 18. If no Background is advertised, the server uses an internal default background for portrayal.

AvailableLODScheme

The AvailableLODScheme element describes a set of Levels of Detail (LODs) that can be provided by the advertised Layers (see Figure 6).

In this document, the term LOD refers to the concept of discrete Levels of Detail, meaning that any given geographic feature may have multiple geometric representations. These representations can be considered independent of each other, for the purpose of portrayal.

For instance, a building may be represented a) as simple box geometry, b) as a geometry with additional façade textures, c) as a group containing elements for walls, roofs, windows, doors, or d) even as a group containing the complete room interior. Each of these representations describes the same geographic feature and can therefore be stored in the same layer. However, it is not necessary that all LODs are consistently available for each feature. A client may assemble its scene graph from subsets of the same layer having different LODs and thus adjust the workload placed on the graphics pipeline.

Usually LODs are organized in one or more LOD schemes, which describe attributes of the individual LODs so a client is able to process and use them adequately. Each LOD scheme (of which there may be one or more) contains a title, abstract, unique identifier, and a full order of LOD definitions, which carry the actual numeric value or magnitude of the LODs.

This LOD value is a URI consisting of the LOD scheme's identifier value as prefix and the actual numeric value, separated by a colon, e.g., ``CityGML:4" for CityGML indoor building models. The prefix indicates the spectrum of possible values and how these values should be interpreted and, conventionally, is fixed per AvailableLODScheme node.

The exact meaning of individual LOD levels remains out of scope for the purposes of this standard. However, the order should, on average, reflect the complexity of members of a certain level of detail definition.

The numeric value indicates the actual `level'' of detail on that ordinal scale. The scale values have a total order, which is connex (a > b or a < b or a = b) and transitive (a > b > c implies a > c), but no interval or metric may be derived from the values. For instance,` CityGML:4" is more accurate than ``CityGML:2", but not necessarily twice as accurate.

The order of the LOD nodes within the AvailableLODScheme node is not defined, but it is recommended to use an order increasing by the LODValue, from lower to higher levels of detail.

Table 21 lists LOD names associated with well-known LOD schemes whose meaning should be preserved, i.e., no conflicting names or semantics should be introduced by a service implementation.

NOTE The commonly understood LOD definitions of CityGML can be found in [OGC CityGML] Subclause 6.2, and are specified in Table 21.

Table 21. Well-known LOD names

LOD names	Description
CityGML:0, CityGML:1, CityGML:2, CityGML:3, CityGML:4	[OGC CityGML] Subclause 6.2

LOD names	Description
INSPIRE:0, INSPIRE:1, INSPIRE:2, INSPIRE:3, INSPIRE:4	same as CityGML

AvailableLODSelection

The AvailableLODSelection components list the LOD selection methods that a client can apply for telling the service how to interpret the requested LOD value for each layer.

Four selection methods are predefined that fit the well-known LOD-based multiple representations approach. These predefined LOD selection methods are defined in Table 22. A specific 3DPS server may define additional LOD selection methods.

Table 22. 3DPS LOD selection methods

Name	Description
equals	For each feature, the available LODs are compared with the requested LOD. If the requested LOD is available for this feature, then the according model shall be selected and included for portrayal response. If the requested LOD is not available for this feature, then the feature shall not be included in the request. This is the default selection method.
equals_or_smaller	This method causes the service to compile a scene from multiple available LODs. Only one LOD for each feature shall be selected. If the requested LOD is available for this feature, then the according model shall be selected and included in the scene. If the requested LOD is not available for this feature, then the service shall select the next lower LOD available for this feature. If no LOD equal or lower than the requested LOD is available for this feature, then this feature shall be omitted in the scene altogether. Cumulative LOD models, e.g., building blocks representing multiple buildings as a single geometry, shall be handled so that no overlaps with higher LODs occur. If a higher LOD for one building included in the block is available, then the block shall be omitted.
combined	This method causes the service to include multiple LODs for each feature in the scene, if available. The service shall include for each feature all LODs equal or smaller than the requested LOD value in the AbstactGetPortrayal request.
equals_or_similar	The service shall make a best effort to find models closely matching the requested LOD. Completeness of the result, if possible free of doubly represented features, is the quality criterion for this strategy.

DeliveryOption

A delivery option is an optional mode that affects how the implementation serves its data. For example, whether the service sends a full textured building or some down-scaled variant that looks alike from a distance may be controlled using a delivery option.

The goal of delivery options is to be able to broker the best-performing exchange mode between a client and a service instance without causing malfunction due to interoperability issues. If an implementation has special features geared towards specific clients, e.g., an optimized streaming option for terrain data, it should use the delivery option as a marker on the layers that support the feature. Clients are expected to possess a positive list of delivery options they support, and to only ask for delivery options they are ready to support.

Delivery options are a way to safeguard tweaks, optimizations, and other means necessary for performance but detrimental to interoperability. Delivery options help to establish interoperability and improve performance in more complex settings, e.g., involving multiple service instances, by making communications more transparent, predictable, and thus dependable.

Interoperability considerations:

Similar to the Format parameter, availability of delivery options may affect the potential for interoperability with a given client. Unlike the Format parameter, delivery options may co-exist in a single request, they may or may not not be subject to interoperability considerations, and finally, delivery options do not require (but benefit from) a shared understanding of the available options. Accordingly, some delivery options may be sensible only in specific clients, uncommon format profiles, or depend on other specific circumstances that a client does not know about.

Clients should match the service-provided list of delivery options with an internal list of desired mechanisms, and generally refrain from requesting unsupported or unwanted delivery options. Delivery options may have an impact on interoperability, but there seems to be no general way of communicating the pitfalls or benefits associated with them. This standard therefore just specifies name and identifiers, which hopefully serve to safely determine the (absence of) potential for interoperability.

Discussion:

The intent behind delivery options is to provide safe means of establishing interoperability. That is, it should be known in advance and not through user-observed or silent failure, if a client may communicate and portray properly data from a set or subset of service instances. MIME types, formats, and their profiles alone are not well-suited to capture the fast-paced evolution in 3D portrayal in the detail required to establish interoperability. In lieu of a commonly accepted mechanism to do that, delivery options enable safeguarding the development of improved mechanisms so that interoperability can be safely determined by machines.

The same goals could be achieved by using MIME types and parameters (e.g., RFC 2231), but while MIME has interoperability as a goal, performance or bandwidth are not generally seen as a concern for MIME. Moreover, many of the practically working approaches encompass several formats, making the reliance on MIME types artificial.

Given these considerations, the ``delivery options'' approach seems a much more workable way of addressing the current diversity of mechanisms. Interoperability may be assessed by humans and can be implanted (in a backwards-compatible manner) into clients because each delivery option is associated with multiple URIs treated as aliases, some of which may serve backwards compatibility. Thus, implementers of this standard may safely evolve their designs as long as they make breaking changes visible through delivery options.

ViewpointHint

A ViewpointHint parameter describes a meaningful camera specification, which a client can use to request a specific portrayal. For this, it suggests meaningful combinations of camera position (POC), camera look-to (POI), camera up-direction (UP) as well as a field of view angle in x-direction (FOVX).

SupportsBoundingBoxConversion

The SupportsBoundingBoxConversion parameter is ``true'' if the service instance supports

conversion of bounding boxes from the request CRS into appropriate layer CRS for query.

Extensions

The Extensions component is provided as a canonical place for extensions and can be used, e.g., as a hook for operation-specific service metadata by 3DPS extension modules.

8.2.3. GetCapabilities exceptions

If a 3DPS service encounters an error while performing a GetCapabilities operation, it **shall** return an exception report message as specified in [OWS Common] Subclause 7.4.1.

8.3. Binding for the GetCapabilities operation

8.3.1. GetCapabilities request HTTP/GET + KVP encoding

The GetCapabilities request HTTP/GET + KVP encoding is as specified in Subsection 7.2.3 of [OWS Common].

EXAMPLE: A GetCapabilities request may look like this:

> http://www.example.com/3dps?SERVICE=3DPS&REQUEST=GetCapabilities&ACCEPTVERSIONS=1.0

8.3.2. GetCapabilities response XML encoding (mandatory)

As desired by [OWS Common], a 3DPS server **shall** offer the service metadata at least in XML format. An XML schema fragment for a service metadata document extends OWS CapabilitiesType in owsCommon.xsd of [OWS Common] as refined for the 3DPS, and may be reviewed under Annex B: XML Schemas (normative).

As indicated, this XML schema document uses the owsServiceIdentification.xsd, owsServiceProvider.xsd, and owsOperationsMetadata.xsd schemas specified in [OWS Common]. It also uses XML schema documents for the Contents and PortrayalCapabilities sections of the service metadata document, which are split per target namespace as listed in Table 23. All these XML schema documents contain documentation of the meaning of each element, attribute, and type, and this documentation **shall** be considered normative as specified in Subclause 11.6.3 of [OWS Common].

Table 23. Namespaces and their schema files

Namespace URI	Schema file
http://www.opengis.net/3dps/1.0/core	3dps-core.xsd
http://www.opengis.net/3dps/1.0/scene	3dps-scene.xsd
http://www.opengis.net/3dps/1.0/view	3dps-view.xsd
http://www.opengis.net/3dps/1.0/info	3dps-info.xsd

EXAMPLE: The response to a valid GetCapabilities request may look like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<Capabilities xmlns="http://www.opengis.net/3dps/1.0/core"
 xmlns:core="http://www.opengis.net/3dps/1.0/core"
 xmlns:ows="http://www.opengis.net/ows/2.0"
 xmlns:xlink="http://www.w3.org/1999/xlink"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.opengis.net/3dps/1.0 ../../../schema/3dpResp.xsd"
version="1.0">
  <ows:ServiceIdentification>
    <ows:Title>3DPS Example Implementation</ows:Title>
    <ows:Abstract>A 3DPS Example</ows:Abstract>
    <ows:Keywords>
      <ows:Keyword>3D</ows:Keyword>
      <ows:Keyword>Portrayal</ows:Keyword>
    </ows:Keywords>
    <ows:ServiceType codeSpace="OGC">3DPS</ows:ServiceType>
    <ows:ServiceTypeVersion>1.0</ows:ServiceTypeVersion>
    <ows:Profile>
http://www.opengis.net/spec/3DPS/1.0/extension/scene/1.0</ows:Profile>
    <ows:Profile>http://www.opengis.net/spec/3DPS/1.0/extension/view/1.0</ows:Profile>
    <ows:Fees>none</ows:Fees>
    <ows:AccessConstraints>none</ows:AccessConstraints>
  </ows:ServiceIdentification>
  <ows:ServiceProvider>
    <ows:ProviderName>Fraunhofer IGD</ows:ProviderName>
    <ows:ServiceContact>
      <ows:PositionName>Geographic Information Management</ows:PositionName>
      <ows:ContactInfo>
        <ows:Address>
          <ows:ElectronicMailAddress>geo@igd.fraunhofer.de</ows:ElectronicMailAddress>
        </ows:Address>
      </ows:ContactInfo>
    </ows:ServiceContact>
  </ows:ServiceProvider>
  <ows:OperationsMetadata>
    <ows:Operation name="GetScene">
      <ows:DCP>
        <ows:HTTP>
          <ows:Get xlink:href="http://example.com/3dps?" />
        </ows:HTTP>
      </ows:DCP>
      <ows:Parameter name="Exceptions">
        <ows:AllowedValues>
          <ows:Value>text/xml</ows:Value>
          <ows:Value>application/vnd.ogc.se_xml</ows:Value>
          <ows:Value>application/vnd.ogc.se_blank</ows:Value>
          <ows:Value>blank</ows:Value>
          <ows:Value>errormarker</ows:Value> <!-- custom extension -->
        </ows:AllowedValues>
        <ows:DefaultValue>text/xml</ows:DefaultValue>
      </ows:Parameter>
```

```
      <!-- ... more parameters -->
    </ows:Operation>
    <ows:Operation name="GetView">
      <ows:DCP>
        <ows:HTTP>
          <ows:Get xlink:href="http://example.com/ogc/3dps?" />
        </ows:HTTP>
      </ows:DCP>
      <ows:Parameter name="Exceptions">
        <ows:AllowedValues>
          <ows:Value>text/xml</ows:Value>
          <ows:Value>application/vnd.ogc.se_xml</ows:Value>
          <ows:Value>application/vnd.ogc.se_inimage</ows:Value>
          <ows:Value>application/vnd.ogc.se_blank</ows:Value>
          <ows:Value>blank</ows:Value>
        </ows:AllowedValues>
        <ows:DefaultValue>text/xml</ows:DefaultValue>
      </ows:Parameter>
      <!-- ... more parameters -->
    </ows:Operation>
    <!-- ... more operations -->
  </ows:OperationsMetadata>
  <Contents>
    <Layer>
      <ows:Title>Buildings</ows:Title>
      <ows:Identifier>buildingLayer</ows:Identifier>
      <ows:BoundingBox>
        <ows:LowerCorner>...</ows:LowerCorner>
        <ows:UpperCorner>...</ows:UpperCorner>
      </ows:BoundingBox>
      <AvailableCRS>EPSG:4327</AvailableCRS>
      <AvailableLOD>CityGML:1</AvailableLOD>
      <AvailableLOD>CityGML:2</AvailableLOD>
      <AvailableStyle>
        <ows:Identifier>textured</ows:Identifier>
      </AvailableStyle>
      <Extensions>
        <!-- ... e.g., extension-specific data -->
      </Extensions>
    </Layer>
    <!-- ... more layers -->
  </Contents>
  <PortrayalCapabilities>
    <OverallStyle>
      <ows:Title>Wireframe</ows:Title>
      <ows:Identifier>wireframe</ows:Identifier>
    </OverallStyle>
    <OverallStyle>
      <ows:Title>Dynamic sky</ows:Title>
      <ows:Identifier>dynamicSky</ows:Identifier>
    </OverallStyle>
```

```
    <AvailableLODScheme>
      <ows:Title>CityGML</ows:Title>
      <ows:Identifier codeSpace="http://www.opengis.net/3dps/1.0">
CityGML</ows:Identifier>
      <LOD>CityGML:0</LOD>
      <LOD>CityGML:1</LOD>
      <LOD>CityGML:2</LOD>
      <LOD>CityGML:3</LOD>
      <LOD>CityGML:4</LOD>
    </AvailableLODScheme>
    <AvailableLODScheme>
      <ows:Title>customScheme</ows:Title>
      <ows:Identifier codeSpace="http://example.com">
ExampleCustomLODs</ows:Identifier>
      <LOD>boundingBox</LOD>
      <LOD>simplified</LOD>
      <LOD>full</LOD>
    </AvailableLODScheme>
    <DeliveryOption>
      <ows:Title>TextureAtlas</ows:Title>

<ows:Identifier>http://www.opengis.net/spec/3DPS/1.0/concept/service/scene/deliveryOpt
ions/textureAtlas</ows:Identifier>
      <Format>image/png</Format>
      <Format>image/jpg</Format>
    </DeliverOption>
    <ViewpointHint>
      <ows:Title>View from top</ows:Title>
            <POC><X1>13.4097</X1><X2>52.5177</X2><X3>220.0</X3></POC>
            <POI><X1>13.4087</X1><X2>52.5202</X2><X3>120.0</X3></POI>
            <UP><X1>0.0</X1><X2>0.0</X2><X3>1.0</X3></UP>
    </ViewpointHint>
    <SupportsBoundingBoxConversion>true</SupportsBoundingBoxConversion>
    <Extensions>
      <!-- ... operation-specific extensions -->
    </Extensions>
  </PortrayalCapabilities>
</Capabilities>
```

8.4. AbstractGetPortrayal operation (abstract)

The abstract core:AbstractGetPortrayal operation specifies the commonality between the two represented portrayal approaches, 3D scene-graph delivery and rendered image delivery. The actual operations (serving a particular approach) **shall** be derived from the AbstractGetPortrayal operation by adding required parameters and specifying those not concretized in the AbstractGetPortrayal operation.

This operation is not to be implemented directly.

8.4.1. AbstractGetPortrayal request

A core:AbstractGetPortrayal request **shall** consist of a core:AbstractGetPortrayal structure as defined in Figure 8 and Table 24. This applies only to actual implementations of the request.

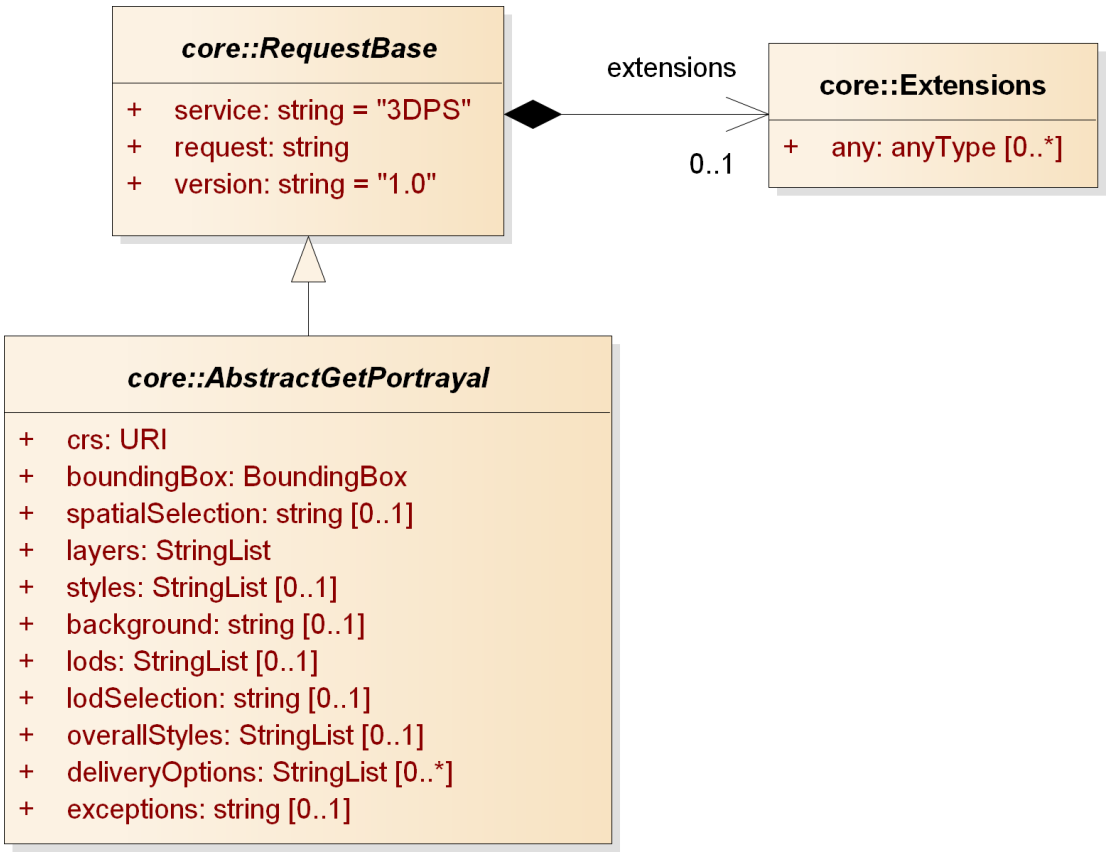

Figure 8. 3DPS core:AbstractGetPortrayal request UML class diagram

Table 24. Components of core:AbstractGetPortayal request

Names	Definition	Data type and value	Multiplicity and use
service Service	Service type identifier	string fixed to ``3DPS''	One (mandatory)
request Request	Operation name	string Value to be specified by concrete operations	One (mandatory)
version Version	Standard version for operation	string, not empty Value is fixed to ``1.0''	One (mandatory)
extensions Extensions	Extension hook	Extensions	Zero or one (optional)
crs CRS	Primary CRS	anyURI as defined in [OWS Common] Subclause 10.3	One (mandatory)
boundingBox BoundingBox	Bounding box corners surrounding selected dataset, in CRS units.	BoundingBox data structure, see [OWS Common] Subclause 10.2.	Zero or more (optional)

Names	Definition	Data type and value	Multiplicity and use
spatialSelection SpatialSelection	Indicates method of selecting objects with BoundingBox	string, not empty. Values are specified in service metadata, see Table 20. Default is ``overlaps''	Zero or one (optional) Include if selection method other than ``overlaps'' is required
layers Layers	List of layer identifiers to retrieve the data from	StringList, not empty Values are specified in service metadata	One (mandatory)
styles Styles	List of one style identifier per requested layer	StringList, not empty Values are specified in service metadata	Zero or one (optional)
background Background	Identifier of desired background	string, not empty Values are specified in service metadata	Zero or one (optional) Include when background desired
lods LODs	List of one LOD identifier per requested layer	StringList, not empty Values are specified in service metadata	Zero or one (optional)
lodSelection LODSelection	Indicates method for selecting LODs	string, not empty Values are specified in service metadata, see Table 22. Default is ``equals''	Zero or one (optional) Include if method other than ``equals'' is required
overallStyles OverallStyles	Identifier(s) of desired overall scene style(s)	StringList, not empty Values are specified in service metadata	Zero or one (optional) Include when overall styling desired
deliveryOptions DeliveryOptions	Identifier(s) of delivery options requested by the client	StringList, not empty Values are specified in service metadata	Zero or more (optional)
exceptions Exceptions	Format of exceptions	ows:MimeType, see [OWS Common] Subclause 10.5	Zero or one (optional)

CRS

The CRS parameter defines the coordinate reference system that is considered to be in effect for the operation request and response. In particular, it is considered as the CRS for the bounding box if that is not specified explicitly. See Coordinate systems for details.

The parameter value for the coordinate reference system (CRS) is defined in Subclause 10.3 of [OWS Common] and [OGC 04-046r3]. If a 2D CRS is used, the height reference is taken from the layer's vertical datum.

BoundingBox

The BoundingBox parameter allows a Client to request a particular spatial subset.

It defines the coordinates of the bounding box corners in at least 2 dimensions and, optionally, the CRS in which they are specified. The CRS given is relevant only to the query and defaults to the one given with the CRS parameter.

If a 3DPS service supports a bounding box CRS other than the available Layer CRS, this capability should be advertised in the metadata document by setting the SupportsBoundingBoxConversion.

The BoundingBox data structure is defined in [OWS Common] Subclause 10.2. The units, ordering, and direction of increment of the x1, x2, and x3 axes are as defined by its CRS element.

If the BoundingBox values are not defined for the given CRS (e.g., latitudes greater than 90 degrees in CRS:84), the service **shall** treat this as an error. If a request contains an invalid bounding box (e.g., one whose minimum x1 is greater than the maximum x1) the service **shall** treat this as an error. No axis wrap-around behavior should be assumed.

SpatialSelection

The SpatialSelection parameter defines the method that a server **shall** use to check the spatial relation between the requested BoungindBox and the features' 3D geometries. Available methods for spatial selection are advertised in the server's metadata document. See AvailableSpatialSelection for more details.

Layers

The Layers parameter specifies a comma-separated list of data layers to be displayed. The concept of the layer is a metaphor to the traditional (two-dimensional) cartography, with which geo objects of different classes were drawn on different transparent foils resulting in a map with an overall view of these foils.

The definition of a layer for the purposes of this standard is lent from WMS: ``basic unit of geographic information that may be requested as a map from a service''.

The Layer parameter contains a list of layer identifiers that specifies the layer to include. The order in which the layers are listed in the Layers parameter does not influence the visual appearance of the generated scene or view. However, the order of the lists in the (optional) Styles and LOD parameters shall correspond with the Layers list. Each entry in the Layers list **shall** refer to a layer identifier as described in the service metadata.

The Layers parameter can be empty, which means not to constrain the output to any specific set of layers. A service receiving an empty Layers parameter could a) serve all data available, b) serve only a subset of the data, or c) serve no data at all.

Styles

The Styles parameter lists the visual styles in which each layer is to be rendered. There is a one-to-one correspondence between the values in the Layers parameter and the values in the Styles parameter: The **Styles** list **shall** contain one Style identifier for each Layer in the Layers parameter;

the order **shall** be the same. Thus, the Styles list **shall** have the same length as the Layers list.

A client may request the default style for a layer using an empty value. If several layers are requested with a mixture of named and default styles, the Styles parameter **shall** include empty values between commas (as in `STYLES=style1,,style2,,'')` to represent default Styles. If all layers are to be shown using their default styles, a request **shall** contain either multiple comma-separated values one for each layer (as in STYLES=,,,'') or a single empty value (``STYLES='').

Each data layer in the list of Layers is rendered using the corresponding style in the same position in the Styles list. Each style name shall be one that was defined for or inherited by this layer as specified in the service metadata. (In other words, the client may not request a layer in a style that was only defined for a different layer.)

If a service advertises several styles for a layer and the client sends a request for the default style, the choice of which style to use as default **shall** be indicated in the service metadata, see Table 13.

Currently, the 3DPS only supports service-defined styles, advertised by identifier in the service metadata. Thus, styling of features may not be transparent to a client. Therefore, it is recommended to include a detailed style description in the service metadata. In case of user styles included in the GetScene request, the client has full control over the styling.

EXAMPLE: Layers=dtm,vegetation,buildings&Styles=orthophoto,,textured

Background

The Background parameter tells the server which background to apply to creating the 3D portrayal. It contains of the background identifiers advertised in the service metadata document.

LODs

The parameter LODs specifies for each layer which Level of Detail (LOD) to choose from when accessing the service's data repository. The parameter value is a list of names referring to the available LODs as specified in the service metadata's PortrayalCapabilities and Contents sections, see AvailableLODScheme.

The length of the LODs list **shall** be equal to the length of the list in the Layers parameter. The order of the LODs list entries shall correlate with the Layers list entries, meaning that LOD *n* **shall** be selected from layer *n*.

LODSelection

In conjunction with the LOD parameter, the LODSelection parameter may be used for telling the service how to interpret the LOD value for each requested layer. The selection methods offered by a 3DPS service are advertised in the service metadata. See AvailableLODSelection for more details on LOD selection methods.

OverallStyles

The OverallStyles parameter specifies, which styles to apply to the overall 3D scene or view. It contains a list of identifiers of OverallStyles as described in the service metadata; see OverallStyle.

If no OverallStyles are specified in the service request, a service implementation should not apply or deliver any. If the combination of styles requested cannot be delivered, the service should make an effort to prioritize the first-mentioned overall styles, or return an exception.

DeliveryOptions

The DeliveryOptions parameter specifies a list of delivery options requested by the client. The client is expected to be prepared to properly handle each of the delivery options. See DeliveryOption for details.

This list is to be interpreted as a request, i.e., failure to use a specific delivery option is not to be treated as an error. The reason is that it is hard to foresee, properly describe, and implement the constraints that might hold for a particular delivery option, e.g., a terrain streaming method might have legal constraints depending on the effective legislation. It is not, in full generality, possible for a client to only ask for delivery options that will work for the service instance. A more mundane case might be a delivery option that is only available for a part of the layers being requested. Treating this as an error would preclude forming such a request.

However, requesting a delivery option that is not advertised for any of the layers being requested, or not at all, should be treated as an error with the DeliveryOptionNotDefined exception code.

Exceptions

The Exceptions parameter specifies the behavior of the service upon detecting an error, e.g., an invalid request or an internal server error. The default value is ``text/xml''. The value shall be one of the MIME types offered in the service metadata document.

If the Exceptions parameter is set to `blank'' (i.e., the character sequence comprised of `b', `l', `a', `n', `k'), then the service **shall**, upon detecting an error, return a document of the MIME type specified in the format parameter whose content is uniformly` off'', i.e., a response document with a valid structure according to the requested format and with no or no useful content. This silent mode is useful if the client is not prepared to process service exceptions. For example, the client may be a generic client that understands the format expected but not OGC exception reports.

Accordingly, the service may issue an HTTP status code of 200 (OK) or 204 (no content) instead of an error code (see AbstractGetPortrayal exceptions).

Extensions

The Extensions parameter carries ancillary informatino to be sent from client to server.

8.4.2. AbstractGetPortrayal response

The AbstractGetPortrayal response **shall** be specified by actual implementations of this abstract operation without restrictions by the abstract operation.

8.4.3. AbstractGetPortrayal exceptions

If a 3DPS service encounters an error while performing a concrete AbstractGetPortrayal operation, it **shall** return an exception report message as specified in Clause 8 of [OWS Common].

In case of an incorrect request or an intermittent error while generating or delivering the response, the response **shall** be supplied in the requested exception format (i.e., the request parameter Exception) by the service. The rules of the underlying DCP are to be observed, in particular the HTTP status code and MIME type should be set according to the exception.

All exception codes defined in [OWS Common] remain valid. The exception codes listed in Table 25 are defined for more specific cases.

Table 25. Additional AbstractGetPortrayal exceptions

exceptionCode value	Meaning of code	locator value
StyleNotDefined	an unadvertised Style is requested	the style name
BackgroundNotDefined	an unadvertised Background is requested	the background name
DeliveryOptionNotDefined	an unadvertised DeliveryOption is requested	the delivery option name
LODNotDefined	an unadvertised LOD name is requested	the LOD name
LODNotApplicable	an unsuitable LOD name is requested	the LOD name

8.5. GetResourceById operation (optional)

Optionally, a 3DPS implementation may offer a GetResourceById operation to deliver resources deemed necessary for operation. The operation follows the blueprint given in [OWS Common] Subclause 9.3, subsequently the ``base GetResourceByID operation''. [2: Following the notion that 'Id' is an abbreviation not an acronym, the preferred casing in this document is 'GetResourceById' not 'GetResourceByID' and 'ResourceId' not 'ResourceID', respectively.]

A typical reason to implement GetResourceById is streaming, i.e., delay-loading pre-computed parts of the data on offer to support interactive users exploring the data. Other types of resources could be textures, geospatial indexes, or parts of a web application to access the service.

There is no obligation to route such resources through a GetResourceById operation, however, it might be preferable for deployment reasons, e.g., to control (re-)generation of resources or to state their origin in sensitive scenarios.

8.5.1. Obtaining ResourceId URIs

There are several ways in which resource identifiers may be obtained, which in principle can be *direct* or *indirect*. *Direct* refers to the specification of a URI suitable to submit as the ResourceId. *Indirect* refers to the specification of a URL that constitutes a valid GetResourceById operation request when resolved and executed.

The only direct way of obtaining resource identifiers is as part of a layer's metadata. It should be noted that, since the output format is required, the direct specification only makes sense when the OutputFormat parameter is also specified.

Indirect resource identifiers may be obtained as URIs embedded in GetCapabilities, GetScene or any other 3DPS operation response, including HTTP redirects where permissible.

No service instance should expect a client to come up with or modify resource identifiers by itself.

8.5.2. Categories of resources

The GetResourceById operation may respond to requests with a variety of resource categories, and this operation does not impose a limit on the categories. However, as called for in the base GetResourceByID operation, it lists typical categories and associated MIME types. An implementation may deliver other resource categories as it sees fit.

Table 26. Examples of resource categories as delivered by the GetResourceById operation.

Type	Example use	Example MIME type(s)
Images	3D views or textures	image/png, image/jpeg
Indexes	Geospatial indexes	application/vnd.google-earth.kml+xml, model/x3d+xml
Geodata	2D/3D geospatial data, tiles or else	application/vnd.google-earth.kml+xml, model/x3d+xml

8.5.3. GetResourceById request

The GetResourceById request strictly follows the base GetResourceByID request with changes and clarifications listed here.

Requirement 3: http://www.opengis.net/spec/3DPS/1.0/req/service/core/getResourceById/request
A core:GetResourceById request **shall** be in conformance with the base GetResourceByID operation request as defined in Subclause 9.3 of [OWS Common], with the additional constraints laid out in this section and in Figure 9 and Table 27.

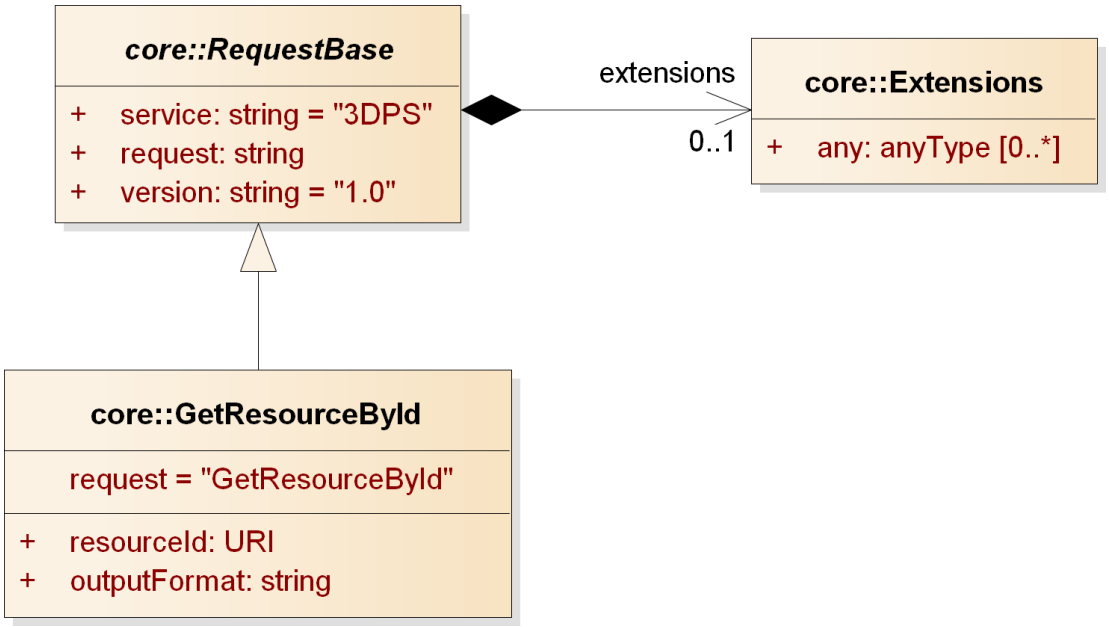

Figure 9. 3DPS core:GetResourceById request UML class diagram

Table 27. Additional components of core:GetResourceById request

Names	Definition	Data type and value	Multiplicity and use
request request	Operation name	string, not empty Value is fixed to ``GetResourceById''	One (mandatory)
extensions Extensions	Container for any kind of ancillary information to be sent from client to server	Extensions type	Zero or one (optional)
resourceId ResourceId	Unambiguous identifier of desired resource	URI, not empty	One (mandatory)
outputFormat OutputFormat	Reference to format in which operation output data should be encoded	string, not empty	One (mandatory)

ResourceId

A ResourceId parameter **shall** be part of each core:GetResourceById operation request.

OutputFormat

The OutputFormat parameter specifies the expected output format. While the output format may be determined by the resource identified with the resource identifier, specifying the output format works in cases where that is not the case (e.g., multiple representations of one resource) but not vice versa. The format, even if not necessary for resource selection, should be used for validation.

8.5.4. GetResourceById response

Requirement 4: http://www.opengis.net/spec/3DPS/1.0/req/service/core/getResourceById/response
The GetResourceById response shall be a document conforming to the requested MIME type, or an exception.

8.5.5. GetResourceById exceptions

Exceptions shall be handled according to the base GetResourceByID operation as described in Subclause 9.3.3.2 of [OWS Common].

Chapter 9. Scene Extension

9.1. Introduction

The GetScene operation of the *Scene* Extension allows a client to retrieve a 3D scene, i.e., graphical data (including geometry and texture data as well as hierarchies), in a standard data format, from a 3DPS service. To achieve an actual 3D portrayal, this data needs to be rendered at the client side. This has the benefit that the client can deliver a responsive navigation experience to the user because it can avoid or delay round-trips to the service instance in most cases.

The GetScene operation is the entry point for a scene-based client to request the geodata that a 3DPS instance is capable of serving in a particular bounding box. At this point, the client is assumed to have issued a GetCapabilites request, processed the response, and determined the instance's capability to serve compatible 3D geodata.

Existing and evolving 3D model formats and their delivery mechanisms vary a lot. Thus, the 3DPS GetScene operation is intentionally defined to allow for a range of mechanisms, potentially limiting the scope for interoperability to quite specific client/server combinations. However, the 3DPS GetScene operation intends to make it possible to integrate different 3D data pools in a single client view. It fosters this interoperability scenario by enabling the client to determine whether the data served by the instances may be combined sensibly at all. In other words, GetScene cannot make 3D data interoperable, but it can help getting there or see why it will not work - before failing horribly in front of the user.

A 3DPS client needs to support the 3D model format, in which a 3DPS delivers a scene as GetScene response. Potential interoperability problems arising from a mismatch of client format capability and implicit service expectations should be dealt with using the content of the GetCapabilities response (in particular delivery options), format-side provisions such as profiles or additional service parameters, in order of preference.

9.2. Modifications to service capabilities

9.2.1. Modifications to ServiceIdentification

A service announces support of the *Scene* Extension to a client by adding the URL identifying this extension to the list of supported extensions delivered in the service metadata document.

Requirement 5: http://www.opengis.net/spec/3DPS/1.0/req/service/scene/extension-identifier
A 3DPS service implementing conformance class *scene* of this *Scene* Extension **shall** include the following URI in a Profile element of the ServiceIdentification in a GetCapabilities response: http://www.opengis.net/spec/3DPS/1.0/extension/scene/1.0.
Dependency: 3DPS Core (http://www.opengis.net/spec/3DPS/1.0/conf-class/core)

9.2.2. Modifications to OperationsMetadata

Requirement 6: http://www.opengis.net/spec/3DPS/1.0/req/service/scene/operations-metadata-getscene
A 3DPS service implementing conformance class *scene* of this *Scene* Extension **shall** include an Operation element in the OperationsMetadata section in a GetCapabilities response having its name attribute set to ``GetScene''.
Dependency: 3DPS Core (http://www.opengis.net/spec/3DPS/1.0/conf-class/core)

9.2.3. Additions to Layer structure

Requirement 7: http://www.opengis.net/spec/3DPS/1.0/req/service/scene/layer-extension
A 3DPS service implementing conformance class *scene* of this *Scene* Extension **shall** extend the core:Layer Extensions structure by zero or one SceneLayerExtension structure as defined in Figure 10 and Table 28.
Dependency: 3DPS Core (http://www.opengis.net/spec/3DPS/1.0/conf-class/core)

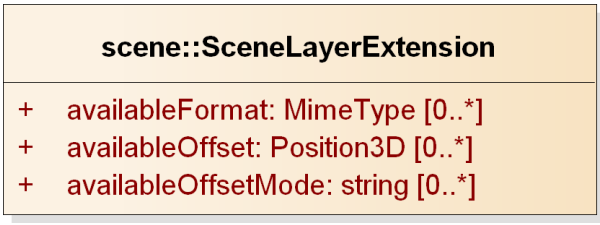

Figure 10. 3DPS scene:SceneLayerExtension UML class diagram

Table 28. Components of scene:SceneLayerExtension structure

Names	Definition	Data type and value	Multiplicity and use
availableFormat AvailableFormat	Output format valid for this layer	ows:OutputFormat type	Zero or more (optional)
availableOffset AvailableOffset	Offset available for querying this layer	Position3D structure, see Table 7	Zero or more (optional)
availableOffsetMode AvailableOffsetMode	Offset mode available for querying this layer	string Values are defined in service metadata	Zero or more (optional)

AvailableFormat

The AvailableFormat element provides information about data formats in which the advertised Layers are available.

AvailableFormat items advertise available encodings of scene data returned by the GetScene operation as ows:MimeType as per [OWS Common] Subclause 10.5.

Table 29 specifies canonical MIME types. These MIME types **shall** be advertised and recognized by a service implementation if the formats represented by them are supported, even if more qualified alternatives exist.

Annex D describes the X3D profiles and nodes that are appropriate for use by a service that returns X3D. It highlights the use of the X3D Geospatial component, which is important to support, as it allows for combination of content from different services.

Table 29. Canonical MIME types for scene encoding (non-exhaustive list)

Encoding format	MIME type
X3D VRML	model/x3d+vrml
X3D XML	model/x3d+xml
OGC KML as XML	application/vnd.google-earth.kml+xml
OGC KML as KMZ	application/vnd.google-earth.kmz
VRML	model/vrml
Esri i3s	application/vnd.esri.i3s.json+gzip
glTF	model/vnd.gltf+json

Requirement 8: http://www.opengis.net/spec/3DPS/1.0/req/service/scene/mimetypes
A 3DPS service implementing conformance class *scene* of this *Scene* Extension **shall** advertise available output formats as MIME types, and additionally **shall** advertise and recognize the MIME types named in Table 29 if the formats represented by them are supported, even if more qualified alternatives exist.

9.2.4. AvailableOffset

The AvailableOffset elements provide information about fixed offsets in which the advertised Layers are available. A client should not expect other offsets to be available unless the SupportsArbitraryOffset portrayal capability is ``true''.

9.2.5. AvailableOffsetMode

Depending on the requested CRS, coordinate values of 3D objects may become bigger than typical 3D display pipelines can handle. For example, UTM coordinates require a mantissa of 9 digits for achieving accuracy in centimeters. Clients using single precision floating point numbers (32 bit; IEEE Std 754-2008) are not able to handle such coordinates very well.

The offset parameter can be used in order to define a reference point in 3D which is then used to enhance the technically available precision, e.g., by subtracting it from all coordinate values, thus reducing the number of significant digits. This offset needs coordination.

Some 3D formats have built-in capabilities to handle geocoordinates, but even in such cases it may be advantageous to use a common offset the client determines. The offset may be applied in several ways, exemplified by the offset modes given in Table 30. For all offset modes it holds that CRS coordinate order is observed in the request. An implementation should choose a suitable offset mode if an explicit mode is not given. An implementation should advertise the offset modes it supports and is free to add more specific offset modes if available. It should support one of the modes given in Table 30 for increased interoperability.

Table 30. Offset modes

Identifier	Definition
subtract	The offset is subtracted from all coordinates specifying absolute geolocations. Thus, when the client adds the offset to them, true geocoordinates in the request CRS are obtained.
embed	The offset is embedded in the resulting scene by means of the request format. Not all formats have such provisions, the client should know if it is capable of handling the request format's appropriate mechanism. It is unspecified whether the offset is actually subtracted from geocoordinates, or whether it is embedded in the request CRS. However, `embed'' should enable the client to obtain true geocoordinates without remembering the offset separate from the result. A possible realization of` embed'' is the X3D ``GeoOrigin'' node.

Under circumstances, the offset may be service-defined, so only one of the offset values offered in the capabilities may be retrieved successfully. If the service advertises offsets, they should include an explicit mode specification.

Clarification: An offset is conceptually different from false easting/northing. False easting/northing serves to make geocoordinates unique, while offset makes unique global coordinates ambiguous. Thus, an offset is NOT expected to be represented in a CRS definition.

Information about available offset modes is to be added to the metadata document as a operation metadata's and operation's parameter definition.

AvailableStyle

Styles usually affect the symbolization of features, e.g., the materials can be replaced by another material, including diffuse color, reflection properties, and transparency as defined in the style. Styling can also take the feature attribute values as input for distinguishing features of different categories by color. Styling may also apply different symbols to point and line features including geometric primitives, billboards, textures, and complex 3D prototypes. The size of these symbols may be scaled according to feature attribute values. It is recommended to use the Symbology Encoding (SE) and Filter Encoding (FE) as basis for defining service styles (see [OGC 05-077r4] and [OGC 09-026r2]), and extend the capabilities for styling 3D objects (see [OGC 09-042]).

9.2.6. Additions to PortrayalCapabilities structure

Requirement 9: http://www.opengis.net/spec/3DPS/1.0/req/service/scene/portrayalcapabilities-extension
A 3DPS service implementing conformance class *scene* of this *Scene* Extension **shall** extend the core:PortrayalCapabilities Extensions structure by zero or one ScenePortrayalCapabilitiesExtension as defined in Figure 11 and Table 31.
Dependency: 3DPS Core (http://www.opengis.net/spec/3DPS/1.0/conf-class/core)

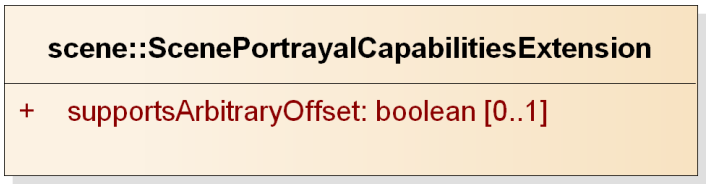

Figure 11. 3DPS scene:ScenePortrayalCapabilitiesExtension UML diagram

Table 31. Components of scene:ScenePortrayalCapabilitiesExtension structure

Names	Definition	Data type and value	Multiplicity and use
supportsArbitraryOffset SupportsArbitraryOffset	Specifies whether the client may request arbitrary offset parameters	boolean type, not empty Value either `true'' or` false''. Default is ``false''	Zero or one (optional)

SupportsArbitraryOffset

The SupportsArbitraryOffset component specifies whether the server allows a client to request arbitrary offset parameters. The element's default value is ``false''.

9.3. GetScene request

Requirement 10: http://www.opengis.net/spec/3DPS/1.0/req/service/scene/getscene/request/structure
A scene:GetScene request **shall** consist of a scene:GetScene structure as defined in Figure 12 and Table 32.

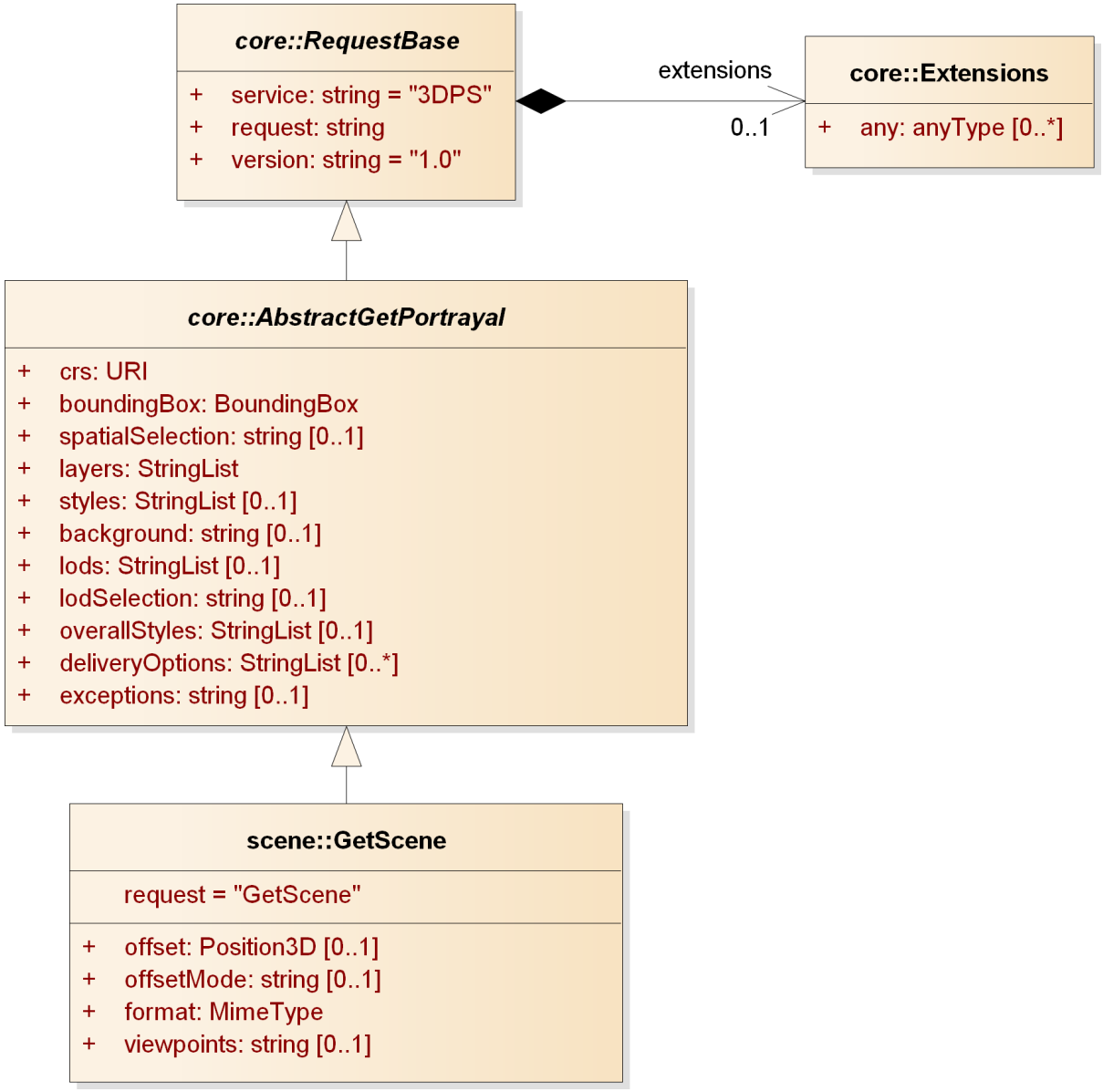

Figure 12. 3DPS scene:GetScene request UML class diagram

Table 32. Additional components of scene:GetScene request

Names	Definition	Data type and value	Multiplicity and use
request Request	Operation name	string Value is fixed to ``GetScene''	One (mandatory)
offset Offset	Offset vector which shall be applied to the scene	Position3D	Zero or one (optional)
offsetMode OffsetMode	Offset mode to use when applying the offset [b]	string	Zero or one (optional)
format Format	Format encoding of the scene	ows:MimeType, see [OWS Common] Subclause 10.5	One (mandatory)
viewpoints Viewpoints	Viewpoints to include in operation response	string, list of tuple values, separated by comma.	Zero or one (optional)

[a] Coordinates are ordered in the request CRS order.
[b] The OffsetMode parameter requires the presence of the Offset parameter.

9.3.1. Offset

The offset to use for scene delivery. See AvailableOffset and AvailableOffsetMode.

9.3.2. OffsetMode

The offset mode to use for applying the offset defined in the OffsetMode parameter. See AvailableOffset and AvailableOffsetMode.

9.3.3. Format

The mandatory Format parameter specifies the target encoding of the returned scene provided as ows:MimeType, see [OWS Common] Subclause 10.5. Available formats are described in the service metadata.

9.3.4. Viewpoints

The Viewpoints parameter can be used to instruct the server to include a list of viewpoints in the scene from which a user can choose from. Viewpoints are presented in most viewers as list which can be used to move quickly to another pre-defined position and view direction.

If multiple viewpoints are defined, then the first in the list **shall** be used as default viewpoint when loading the scene. This is mainly useful to create a HTTP/GET-based link into a 3D scene.

The Viewpoints parameter contains a comma separated list of tuple values defining for each viewpoint a) a Title, b) the Point of Interest (POIx, POIy, POIz), c) the Point of Camera (POCx, POCy, POCz), d) the up vector (UPx, UPy, UPz), and e) the horizontal Field of View (FOVX). Thus, 11 values are used to define one viewpoint. In case of multiple viewpoints to insert, all values are appended to the list consecutively; the length of the Viewpoints parameter list values **shall** be divisible by 11:

```
Viewpoints = Viewpoint,{Viewpoint}
Viewpoint = Title,POIx,POIy,POIz,POCx,POCy,POCz,UPx,UPy,UPz,FOVX
```

All tuple values refer to the spatial reference system specified by the request CRS parameter. The FOV is specified in arc degrees.

9.3.5. Delivery Options

Typical delivery options in GetScene based requests may be obtained from Table 33.

Table 33. Additional DeliveryOption names

Title	Identifier	Description
spatialIndex SpatialIndex	http://www.opengis.net/spec/3DPS/1.0/concept/service/scene/deliveryOptions/spatialIndex	deliver spatially annotated links to this service, e.g., KML NetworkLink, X3D GeoLOD, ESRI i3s nodes, but generally not any actual geometries

Title	Identifier	Description
textureAtlas TextureAtlas	http://www.opengis.net/spec/3DPS/1.0/concept/service/scene/deliveryOptions/textureAtlas	try to recombine textures to optimize rendering
regroupGeometry RegroupGeometry	http://www.opengis.net/spec/3DPS/1.0/concept/service/scene/deliveryOptions/regroup	regroup geometry to optimize rendering; object identity needs not to be maintained
reduceQuality ReduceQuality	http://www.opengis.net/spec/3DPS/1.0/concept/service/scene/deliveryOptions/reduceQuality	Use any technique available to save bandwidth, source data quality needs not to be maintained

9.4. GetScene response

The response to a valid GetScene request is a document of the MIME type as specified in the request Format parameter, except under error conditions. This document contains a 3D scene assembled from the features of the selected Layers (mainly) within the specified BoundingBox, represented in the specified CRS, Format, and Styles.

NOTE This implies CRS coordinate order, which may be very different to an applicable client-side computer graphics convention. It is the client's responsibility to execute any transforms required to convert from CRS axis order to the client's axis order convention.

If the GetScene request contains a list of service style identifiers and/or overall style identifiers, then the features delivered are portrayed according to the respective style(s). Some output formats may not support all the portrayal capabilities (here, this term refers to format provisions not subject to the 3DPS) that are called for by the LODSelection, for defining a virtual camera/viewpoint, or to support certain styles. It is left to the implementation to decide whether to treat this as an error condition or to continue on a best-effort basis. However, if such a condition arises and the implementation chooses to treat it as an error, the error code should be FormatCapabilityMissing. The same holds when the service implementation imposes the shortcoming, not the format definition itself. For example, this is the case when a ``combined" LOD is requested but the format does not have provisions to switch individual features as appropriate.

If the response data contains references to additional resources which are to be loaded by the client, for instance URIs of textures, then those resources should be accessible to the client following standard [RFC 3986] resolution rules, where the base URI is the GetScene request URI that produced the response. Relative paths may be used, but are discouraged due to their potential interference with service invocations.

URIs within the GetScene response might also point to GetResourceById operation requests to the same service or other accessible 3DPS instances, or may point to other service end points producing the required MIME type. For example, a WCS may be used as source for terrain textures. To maximize interoperability, URIs representing service calls should be valid service invocations in themselves, i.e., not require the client to understand the service structure. In other words, service invocations should be treatable as resources. Notwithstanding this, an advanced or special client

could take advantage of prior knowledge about certain services to deliver a better experience. Such behavior should be guarded by appropriate delivery options to make sure a general client does not request it by accident.

Requirement 11: http://www.opengis.net/spec/3DPS/1.0/req/service/scene/getScene/response
The GetScene response **shall** be a document conforming to the requested MIME type, or an exception.

9.5. GetScene exceptions

A GetScene request may return any of the exception reports defined in core:AbstractGetPortrayal and/or as specified in Clause 8 of [OWS Common].

In addition, a service **shall** throw a service exception (code = FormatCapabilityMissing) if an unavailable format-specific feature is requested. The locator should be the parameter containing the unsupported style or other portrayal feature, and the text should be indicative of the root cause.

9.6. Binding Extensions for the GetScene operation

9.6.1. HTTP/GET + KVP binding

The GetScene request may be issued as a HTTP/GET KVP-based request.

Table 34 defines the KVP encoding for the GetScene request. It contains the complete parameter mapping including the parameters defined by core:AbstractGetPortrayal and core:RequestBase.

Table 34. GetScene request KVP encoding

Name and example [a]	Optionality	Definition and format
service=3DPS	Mandatory	Service type identifier
request=GetScene	Mandatory	Operation name
version=1.0	Mandatory	Standard version for operation
crs=epsg:4327	Mandatory	Identifier URI of primary CRS
boundingBox=0.0,0.0,0.0,10.0,10.0,50.0	Optional	Bounding box surrounding selected dataset, in available CRS
spatialSelection=cut	Optional	Method of selecting objects with BoundingBox
layers=layer1,layer2	Mandatory	Identifiers of layers to retrieve the data from, comma-separated list
styles=style1,style2	Optional	Identifier of service style to be used applied, comma-separated list
background=darkSky	Optional	Background identifier as string
lods=citygml:4	Optional	List of LODs requested for the layers, comma-separated list

Name and example [a]	Optionality	Definition and format
lodSelection=equals_or_similar	Optional	Method for selecting LODs
overallStyles=photorealistic	Optional	Identifier(s) of desired overall scene style(s)
deliveryOptions=regroupGeometry	Optional	URIs: Names of the delivery options requested by the client, comma-sparated
exceptions=text/xml	Optional	Format of exceptions
offset=200000,200000,0	Optional	Offset vector which shall be applied to the scene, using the mode specified (if any)
format=model/x3d+xml	Mandatory	Format encoding of the scene
viewpoints=Overview,13.4097,52.5177, 220.0,13.4087,52.5202,120.0,100	Optional	Viewpoints as tuple values to be added by server, comma-separated list of string

[a] All parameter names are listed here using mostly lower case letters. However, any parameter name capitalization shall be allowed in KVP encoding, see Subclause 11.5.2. All example values depend on how a specific OWS specifies them.

Chapter 10. View Extension

10.1. Introduction

The *View* Extension defines the GetView operation, which allows a client to retrieve readily rendered images, i.e., visual representations of an underlying 3D scene, as plain standard images. Image generation is done completely on the server side. This allows for high quality 3D visualization in identical quality on any device, including resource-poor mobile devices.

10.2. Concepts

This section explains the concepts behind the GetView operation. Readers familiar with computer graphics concepts may want to skip it.

10.2.1. Image layer concept

Besides color images, the *View* Extension defines additional image layers of a 3D view, which represent discrete geometric and thematic information for each image pixel. They follow the G-Buffer concept used in computer graphics and are not necessarily meant for human consumption.

It is foreseen to encode non-color image layers using standard image formats as well. This allows for applying the same principles for data encoding, data compression, data exchange, and client-side data loading and processing for all image layers.

Table 35 lists the image layer types that are suggested by the 3DPS *View* Extension and that are described in the following.

Table 35. Image layer names and descriptions

Names	URI	Description
Color	http://www.opengis.net/spec/3DPS/1.0/concept/service/view/imageLayers/color	contains color value for each pixel
Depth	http://www.opengis.net/spec/3DPS/1.0/concept/service/view/imageLayers/depth	contains depth values describing the view-space distance to each pixel
ObjectId	http://www.opengis.net/spec/3DPS/1.0/concept/service/view/imageLayers/objectid	contains object identifier values identifying the features at a pixel
Normal	http://www.opengis.net/spec/3DPS/1.0/concept/service/view/imageLayers/normal	encodes the surface normal for the surface represented
Mask	http://www.opengis.net/spec/3DPS/1.0/concept/service/view/imageLayers/mask	encodes for each pixel, whether any feature is visible there

Image layer types

Color layer

A color layer contains a color value for each pixel, e.g., RGB or RGBA color data. Alpha values are required for storing a transparent background. For color layers, standard image encodings provide good compression results.

Depth layer

A depth layer encodes for each pixel the distance to the visible surface. Unit of measure **shall** be meter (#m) as defined by GML3 [OGC 03-105r1]. This representation abstracts from computer graphical details and the distance values can be used without additional computation in an application. Also, depth images represent a major means to compose multiple images generated from the same camera position and by the same projection.

ObjectId layer

Object identifiers (object-ids for short) are distinct numeric values assigned to all the features in the 3D geo database provided by the 3DPS. An ObjectId layer contains an object-id for each pixel and allows a client application, e.g., to determine all the pixels that show a specific feature or to retrieve feature information based on the object-id.

For a 3DPS service, object-ids **shall** be unique over all provided data Layers and even over multiple GetView requests, due to facilitating consistent interaction across multiple 3D views. The object-id basically refers to the graphical representation of a feature and does not necessarily equal to the feature's identifier. It is rather left to a specific 3DPS service how to determine and assign unique object-ids to all features. Object-id value ``0'' is reserved for pixels that do not represent a feature (e.g., pixels representing the sky) or for which no object-id could be determined.

Normal layer

A normal layer describes for each pixel the direction of the surface normal visible at that pixel. A normal layer might be used, e.g., for computing good camera positions of client-side computation of image effects. The normal layer contains the value ``0'' for a pixel that does not represent any surface point (e.g., sky or transparent background).

Mask layer

A mask layer contains a value of `1'' for each pixel that covers a scene object and` 0'' otherwise. For example, Mask layers provide information about unused image space or support client-side altering of the scene background.

Other image layers

The image layers specified in this *View* Extension represent a fundamental subset of view-related data. However, image layers are not limited to the specified ones. A 3DPS service can implement and advertise additional image layers in the service's metadata document together with available encodings and formats.

Image layer encodings

Image layer encodings can differ in

a. the way of representing image layer data, e.g., as RGB colors

b. the way of encoding this data as standard image.

Image layer encodings/formats **shall** be advertised as (parameterized) MIME types in the service's metadata document.

The *View* Extension suggests representing image layer data in colors and encoding them by appropriate standard image formats. Alternative encodings/formats are possible.

In the following, default encodings for the suggested image layers are described. Table 36 provides an overview of the corresponding MIME types describing these default encodings. A MIME type parameter ``mode'' **shall** be used for adjusting the required bit depth.

Table 36. Image layer encodings (non-exhaustive)

Image layer name	MIME types
Color	image/jpeg
	image/png [a]
	image/png;mode=24bit
	image/png;mode=32bit
Depth	image/png [a]
	image/png;mode=24bit
	image/png;mode=32bit
ObjectId	image/jpeg
	image/png [a]
	image/png;mode=24bit
	image/png;mode=32bit
Normal	image/jpeg
	image/jpeg;mode=24bit
	image/png;mode=32bit
	image/png [a]
	image/png;mode=24bit
	image/png;mode=32bit
Mask	image/jpeg [b]
	image/png [a]
	image/png;mode=1bit

[a] Server chooses bit depth.
[b] Inefficient due to 24-bit data storage.

For alternative service-specific data encodings, a MIME type parameter ``encoding'' should be used for identifying the data encoding and format.

EXAMPLE: A 3DPS service-specific MIME type for a normal layer encoding could be ``image/png;mode=24bit,encoding=two-component'' which specifies the representation of normal vectors by two components only.

Depth layer default encoding

Per default, depth values **shall** be represented by float values. They **shall** be encoded as colors by converting the float value into a Byte array and assigning these Bytes to the color components. The endianness **shall** be BGR for 24-bit encoding or ABGR for 32-bit encoding. Due to the direct storage as color components, resulting depth images possess very little pixel-to-pixel coherence, and should not be stored in lossy image formats.

ObjectId layer default encoding

Per default, object-ids **shall** be represented by unsigned integer values. They are encoded as colors by converting the object-id integer value into a Byte array and assigning these Bytes to the color components. The endianness **shall** be BGR for 24-bit encoding or ABGR for 32-bit encoding. This does not specify the internal endianness of the image format used to encode the colors, rather the mapping of bytes to properly decoded and separated color components.

Normal layer default encoding

Per default, normal layers encode normalized normal vectors in world space by encoding each vector component (X,Y,Z) as color component (B,G,R) of a 24-bit color image. A surface normal **shall** be encoded as 24-bit RGB color value in the following way:

1. Surface normal **shall** be considered in world space.
2. Surface normal **shall** be normalized to length of 1.0.
3. Each normalized normal **shall** be transformed into a right-hand Cartesian coordinate system, having the z-axis pointing up, which leads to a normalized normal n = (x, y, z).
4. For each X, Y, and Z component of the transformed normal a decimal value **shall** be computed by adding 1 and dividing the result by 2.
5. These values **shall** represent the color components, B, G, and R respectively. Endianness **shall** be BGR.

A 3DPS consumer requesting a normal layer encoded in this default format, **shall** decode this data the other way round, by multiplying with 2 and subtracting 1 and assigning the value to the x, y, and z components of a normal vector.

Mask layer default encoding

Mask values 0 and 1 **shall** be color-encoded as follows: white **shall** be used for mask value `0'', i.e., for pixels that do not cover a scene object; the color black shall be used mask value` 1'', i.e., for pixels that do cover scene objects. Mask layers could be encoded in any image format, but formats capable of 1-bit content should be preferred for coding efficiency.

10.2.2. 3D projections

Two-dimensional representations of a 3D virtual environment are generated by transforming points of the 3D scene onto a projection surface. Planar projections can be categorized into perspective projections and parallel projections, which can be further sub-categorized, e.g., in one and two-point perspective projections or orthographic and oblique parallel projections (Figure 13). Besides this, projections can be planar or non-planar. With planar projections, points in the 3D space are linearly mapped to points on a 2D projection plane, i.e., the 3D point, the projected point, and the center of projection are collinear. With non-planar projections, these rays from center of

projection to the three-dimensional point are non-linearly mapped, but are, e.g., projected spherical or cylindrical.

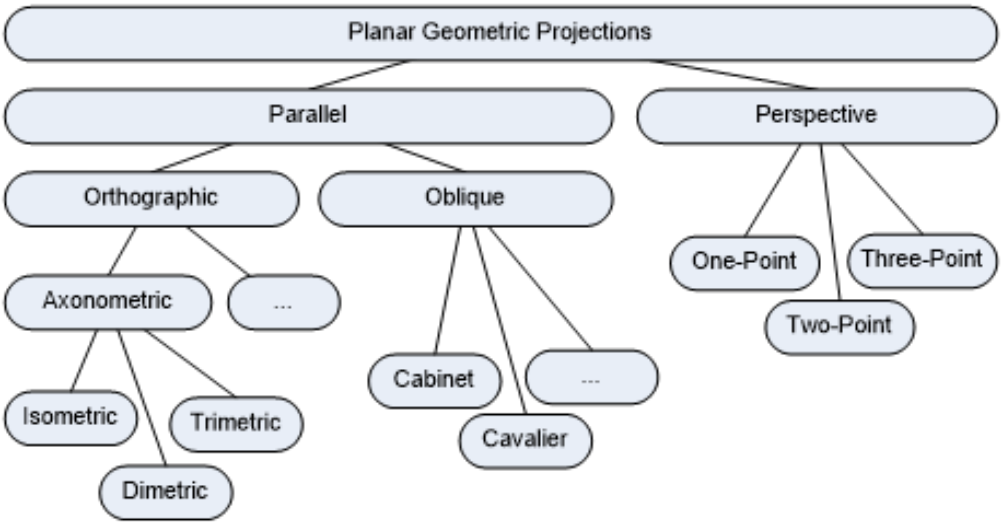

Figure 13. Examples of projection types.

While a 1-point central perspective projection is close to the human perception of the real world, various application domains and visualization techniques could benefit from additional projection types. Because of that, the *View* Extension allows for choosing a projection type that **shall** be applied for rendering the view (Figure 14).

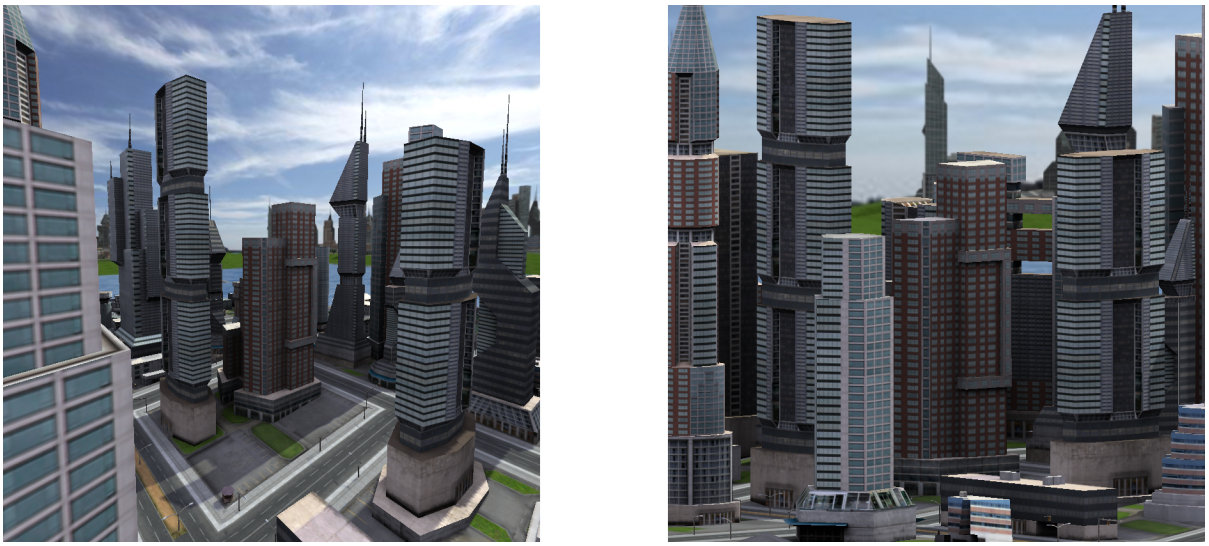

Figure 14. Examples of perspective projection (left) and orthographic projection (right).

The 3DPS *View* Extension specifies two planar projection types, a perspective projection type and an orthographic projection type, Perspective and Orthographic, as defined in Table 37, Figure 15, Table 38, Table 39 and Table 40.

For this, the ProjectionBase data structure forms the basis of specific projections as specified by extensions of this standard.

NOTE	For an XML binding, this standard defines the schema element core:Projection of type core:ProjectionBaseType as the head of a substitution group of projection types.

A specific 3DPS service can provide additional projection types. Supported projection types **shall** be advertised in the service metadata document. Possible values for the parameters of each projection type should be advertised in the service's metadata.

Table 37. Supported projection types and meaning

Identifier	Description
Perspective	PerspectiveProjection as defined in Table 39
Orthographic	OrthographicProjection as defined in Table 40

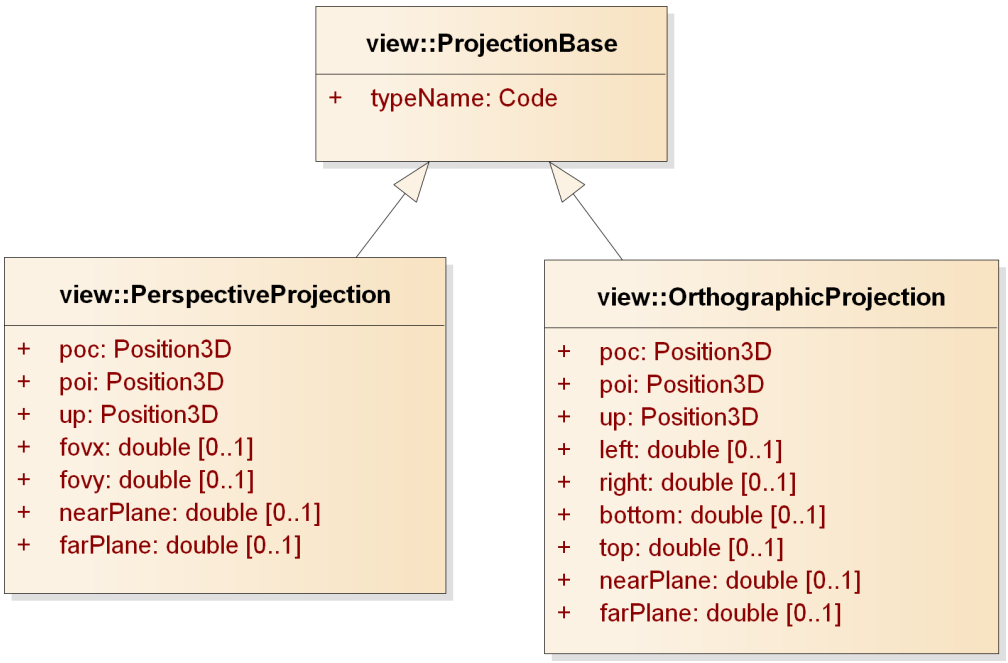

Figure 15. 3DPS view:PerspectiveProjection and core:OrthographicProjection UML class diagram

Table 38. Components of core:ProjectionBase structure

Names	Definition	Data type and value	Multiplicity and use
typeName TypeName	Unambious identifier of this projection type	ows:Code type, not empty	One (mandatory)

Table 39. Components of view:PerspectiveProjection structure

Names	Definition	Data type and value	Multiplicity and use
typeName TypeName	Unambious identifier of this projection type	ows:Code type, not empty fixed to ``Perspective''	One (mandatory)
poc POC	Position of the virtual camera (point of camera)	Position3D type, see Table 7	One (mandatory)
poi POI	Position of point of interest	Position3D type, see Table 7	One (mandatory)
up Up	Position of a point forming the vector pointing in ``up'' direction by subtracting the POC	Position3D type, see Table 7	Zero or one (optional) Include when camera roll desired

Names	Definition	Data type and value	Multiplicity and use
fovx FOVX	Field of view of the virtual camera in horizontal direction	Double type Value in degrees, service metadata shall specify default value	Zero or one (optional) Include when FOVX other than default value desired
fovy FOVY	Field of view of the virtual camera in vertical direction	Double type Value in degrees, service metadata shall specify default value	Zero or one (optional) Include when FOVY other than default value desired
nearPlane NearPlane	Distance to the camera's near clipping plane	Double type Default value is advertised in service metadata. Values in the measure of the request CRS[a]	Zero or one (optional) Include when NearPlane other than default desired
farPlane FarPlane	Distance to the camera's far clipping plane	Double type Default value is advertised in service metadata. Values in measure of the request CRS[a]	Zero or one (optional) Include when FarPlane other than default desired
[a] In the case of a 2D request CRS, measure unit is meter			

Table 40. Components of view:OrthographicProjection structure

Names	Definition	Data type and value	Multiplicity and use
typeName TypeName	Unambious identifier of this projection type	ows:Code type, not empty fixed to ``Orthographic''	One (mandatory)
poc POC	Position of the virtual camera (point of camera)	Position3D type, see Table 7	One (mandatory)
poi POI	Position of point of interest	Position3D type, see Table 7	One (mandatory)
up Up	Position of a point forming the vector pointing in ``up'' direction by subtracting the POC	Position3D type, see Table 7	Zero or one (optional) Include when camera roll desired
left Left	Distance to the left border of the view frustum	Double type Default value is advertised in service metadata. Value in measure of the request CRS[a]	Zero or one (optional) Include when value other than default desired
right Right	Distance to the right border of the view frustum	Double type Default value is advertised in service metadata. Value in measure of the request CRS[a]	Zero or one (optional) Include when value other than default desired

Names	Definition	Data type and value	Multiplicity and use
bottom Bottom	Distance to the bottom border of the view frustum	Double type Default value is advertised in service metadata. Value in measure of the request CRS[a]	Zero or one (optional) Include when value other than default desired
top Top	Distance to the top border of the view frustum	Double type Default value is advertised in service metadata. Value in measure of the request CRS[a]	Zero or one (optional) Include when value other than default desired
nearPlane NearPlane	Distance to the camera's near clipping plane	Double type Default value is advertised in service metadata. Values in the measure of the request CRS[a]	Zero or one (optional) Include when NearPlane other than default desired
farPlane FarPlane	Distance to the camera's far clipping plane	Double type Default value is advertised in service metadata. Values in measure of the request CRS[a]	Zero or one (optional) Include when FarPlane other than default desired
[a] In the case of a 2D request CRS, measure unit is meter			

POC, POI, Up

In PerspectiveProjection and OrthographicProjection data structures, the mandatory POC and POI parameters and the optional Up parameter specify the position and orientation of a virtual camera used for generating the 3D view on the 3D scene.

NearPlane, FarPlane

In PerspectiveProjection and OrthographicProjection data structures, the optional NearPlane and FarPlane parameters specify the distance to the near clipping plane and far clipping plane. A 3DPS service **shall** suggest appropriate default values in the NearPlaneHint and FarPlaneHint elements of the service metadata.

FOVX, FOVY

In PerspectiveProjection data structure, the optional FOVX and FOVY parameters specify the field of view of the virtual camera. Values are in degrees. FOVX specifies the horizontal angle of view, FOVY specifies the vertical angle of view. The service **shall** suggest default values for FOVX and FOVY in the service metadata. If FOVX and FOVY are not provided, the service **shall** use the default values. If only one of FOVX or FOVY is provided, the service **shall** compute the missing value from the aspect ratio of the requested Width and Height.

Left, Right, Bottom, Top

In OrthographicProjection data structure, the optional Left, Right, Bottom, and Top parameters specify the left, right, lower, and upper borders of the cuboidal view frustum. These borders **shall** be specified as distance to the center of projection. The service **shall** suggest default values for Left, Right, Bottom, and Up in the service metadata. If Left, Right, Bottom, and Up are not provided, the

service **shall** use the default values.

EXAMPLE: An orthographic projection resulting in an image having the center of projection (POI) in the center of that image includes a parameter set such as: Left=-100, Right=100, Bottom=-100, Top=100.

10.2.3. Extensibility

The *View* Extension is designed for supporting extensibility of the following aspects of 3D portrayal:

a) Image layers: Beyond image layers COLOR, OBJECTID, DEPTH, NORMAL, MASK, specific 3DPS services can provide additional image layers. Those **shall** be advertised in the 3DPS service's metadata document.

b) Image layer encodings: Beyond encoding image layers as suggested in this specification, specific 3DPS services can provide other encodings. These encodings **shall** be advertised in the 3DPS service's metadata document.

EXAMPLE: A specific 3DPS service could provide an image layer containing only the specular lighting information, which can be used for image post-processing.

10.3. Modifications to service capabilities

10.3.1. Modifications to ServiceIdentification

A service announces support of the *View* Extension to a client by adding the URL identifying this extension to the list of supported extensions delivered in the service metadata document.

Requirement 12: http://www.opengis.net/spec/3DPS/1.0/req/service/view/extension-identifier
A 3DPS service implementing conformance class *view* of this *View* Extension **shall** include the following URI in a Profile element of the ServiceIdentification section in a GetCapabilities response: http://www.opengis.net/spec/3DPS/1.0/extension/view/1.0.
Dependency: 3DPS Core (http://www.opengis.net/spec/3DPS/1.0/conf-class/core)

10.3.2. Modifications to OperationsMetadata

Requirement 13: http://www.opengis.net/spec/3DPS/1.0/req/service/view/operations-metadata-getview <A 3DPS service implementing conformance class *view* of this *View* Extension **shall** include an Operation element in the OperationsMetadata section in a GetCapabilities response having its name attribute set to ``GetView".
Dependency: 3DPS Core (http://www.opengis.net/spec/3DPS/1.0/conf-class/core)

10.3.3. Additions to Layer structure

This *View* Extension does not add any components to the core:Layer structure defined in Figure 6 and Table 12.

10.3.4. Additions to PortrayalCapabilities structure

Requirement 14: http://www.opengis.net/spec/3DPS/1.0/req/service/view/portrayalcapabilities-extension
A 3DPS service implementing conformance class *view* of this *View* Extension **shall** extend the core:PortrayalCapabilities Extensions structure by zero or one ViewPortrayalCapabilitiesExtension as defined in Figure 16, Table 41, Table 42, Table 35, Table 43, and Table 44.
Dependency: 3DPS Core (http://www.opengis.net/spec/3DPS/1.0/conf-class/core)

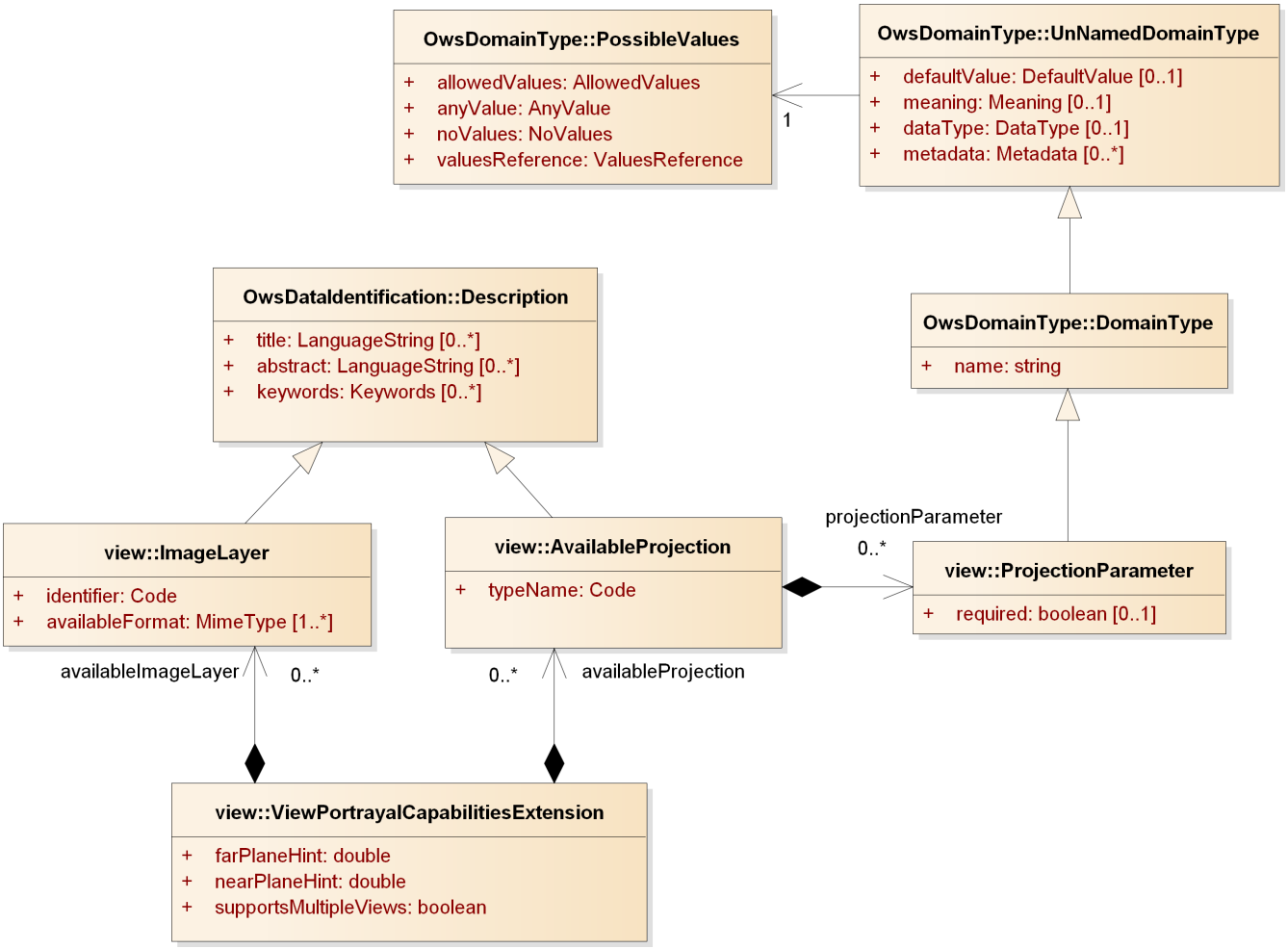

Figure 16. 3DPS view:ViewPortrayalCapabilitiesExtension UML class diagram

Table 41. Components of view:ViewPortrayalCapabilitiesExtension structure

Names	Definition	Data type and value	Multiplicity and use
availableImageLayer AvailableImageLayer	Image layers that can be served by this service	ImageLayer type, see Table 42 At least default ``COLOR'' layer shall be supported	Zero or more (optional) Include when more than the ``COLOR'' layer are supported

Names	Definition	Data type and value	Multiplicity and use
availableProjection AvailableProjection	Specification of a projection that is supported by this service	AvailableProjection type, see Table 43	Zero or more (optional) Include one for each supported projection type
nearPlaneHint NearPlaneHint	Hint at convenient near plane parameter value	PositiveNumber type	Zero or one (optional)
farPlaneHint FarPlaneHint	Hint at convenient far plane parameter value	PositiveNumber type	Zero or one (optional)
supportsMultipleViews SupportsMultipleViews	Signals if the service supports the retrieval of multiple portrayals or image layers	Boolean type Default shall be ``true''	Zero or one (optional) Include when multipart response is not supported

Table 42. Components of view:ImageLayer structure

Names	Definition	Data type and value	Multiplicity and use
title Title	Title of this image layer, human readable	ows:LanguageString, see [OWS Common] 10.7	Zero or more (optional)
abstract Abstract	Brief narrative description of this image layer	ows:LanguageString, see [OWS Common] 10.7	Zero or more (optional)
keywords Keywords	Unordered list of one or more commonly used or formalized word(s) or phrase(s) used to describe this image layer	ows:Keywords type in ows19115subset.xsd	Zero or more (optional) One for each keyword authority used
identifier Identifier	Unambiguous identifier of this image layer, unique for this server	ows:Code type, not empty	One (mandatory)
availableFormat AvailableFormat	Format that is available for this ImageLayer	ows:MimeType, not empty, see [OWS Common] Subclause 10.5	One or more (mandatory) One for each supported image layer format

Table 43. Components of view:AvailableProjection structure

Names	Definition	Data type and value	Multiplicity and use
title Title	Title of this projection, human readable	ows:LanguageString, see [OWS Common] 10.7	Zero or more (optional)
abstract Abstract	Brief narrative description of this projection	ows:LanguageString, see [OWS Common] 10.7	Zero or more (optional)

Names	Definition	Data type and value	Multiplicity and use
keywords Keywords	Unordered list of one or more commonly used or formalized word(s) or phrase(s) used to describe this projection	ows:Keywords type in ows19115subset.xsd	Zero or more (optional) One for each keyword authority used
typeName TypeName	Unambiguous identifier of this projection type, unique for this server	ows:Code type, not empty	One (mandatory)
projectionParameter ProjectionParameter	Parameter of this projection type	ProjectionParameter, which is derived from ows:DomainType, see [OWS Common] Subclause 13.2.1, and has an attribute described in Table 44	Zero or more (optional) Include one for each parameter of the projection

Table 44. Attribute of view:ProjectionParameter structure

Names	Definition	Data type and value	Multiplicity and use
required Required	Flag signaling whether projection parameter is required or can be omitted	Boolean type, not empty Default shall be ``true''	Zero or one (optional) Include when value other than ``true'' is desired

NearPlaneHint, FarPlaneHint

For retrieving good visual results from the 3D portrayal service, the near and far clipping planes restricting the viewing frustum have to be set appropriately. The NearPlaneHint and FarPlaneHint parameters provide a hint on such values.

SupportsMultipleViews

The SupportsMultipleViews parameter is a flag that signals if a 3DPS service supports responding multiple views or multiple image layers by a single GetView response as HTTP multipart/mixed message (see MIME multipart response).

10.4. GetView request

Requirement 15: http://www.opengis.net/spec/3DPS/1.0/req/service/view/getview/request/structure
The view:GetView request **shall** comply with the structure defined in Figure 17 and Table 45.

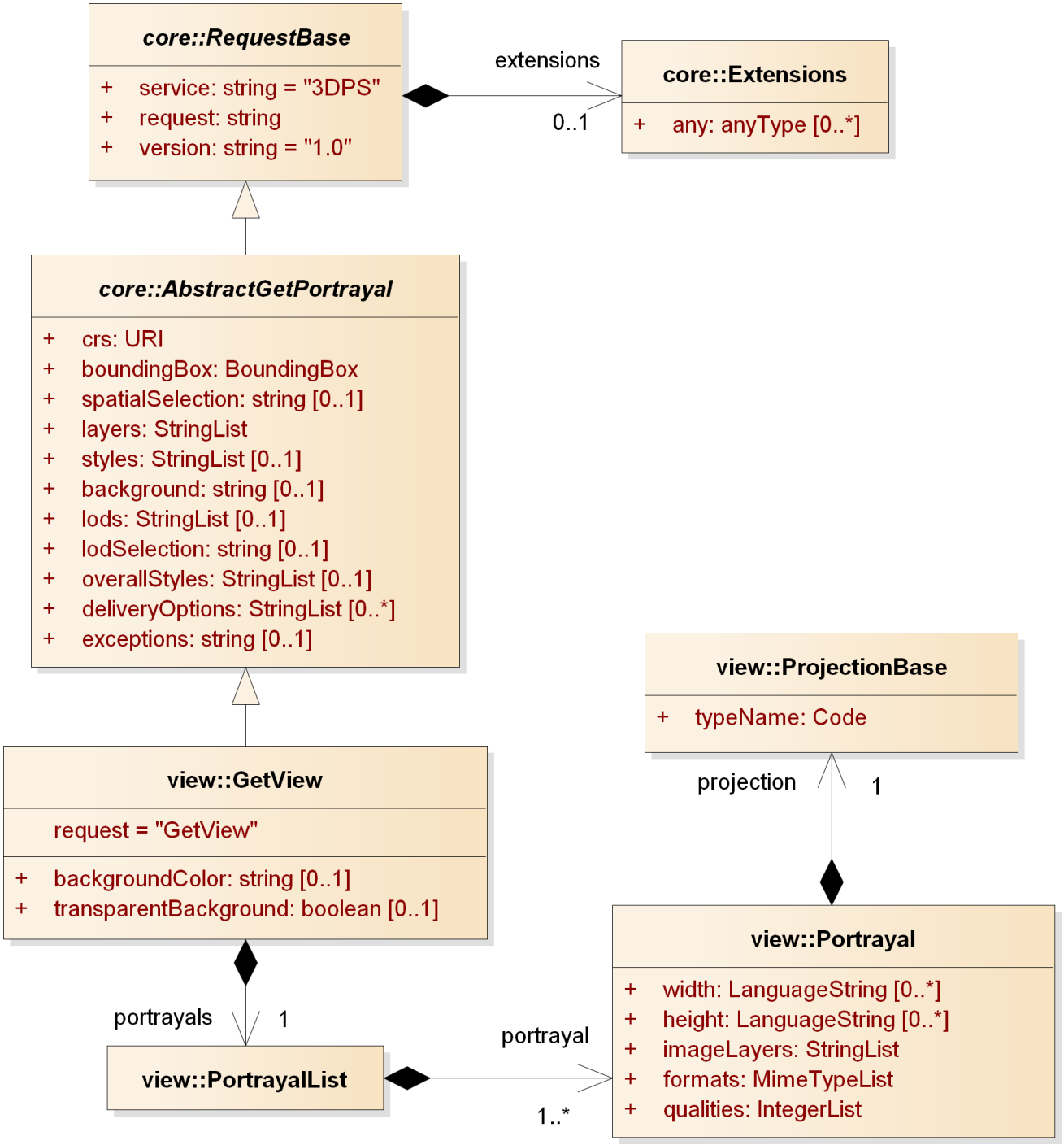

Figure 17. 3DPS view:GetView request UML class diagram

Table 45. Additional components of view:GetView request

Names	Definition	Data type and value	Multiplicity and use
request Request	Operation name	string Value is fixed to ``GetView''	One (mandatory)
backgroundColor BackgroundColor	Background color desired	string, not empty Hexadecimal RGB color values, format 0xRRGGBB. Default is 0xFFFFFF (white)	Zero or one (optional) Include when background color other than white desired

Names	Definition	Data type and value	Multiplicity and use
transparentBackground TransparentBackground	Background transparency desired [a]	Boolean type, not empty Default is ``false''	Zero or one (optional) Include when transparent background desired
portrayals Portrayals	List of portrayal output specifications	List of Portrayal type, not empty, see Table 46	One (mandatory)

[a] When provided within the same request, only that one of the two or three background-related parameters is applied that is evaluated first. Order of evaluation is: Background before BackgroundColor before TransparentBackground.

10.4.1. BackgroundColor

The optional BackgroundColor parameter is a string that specifies the color to be used as the background (non-data) pixels of a COLOR image layer. The general format of BackgroundColor is a hexadecimal encoding of an RGB value where two hexadecimal characters are used for each of red, green, and blue color values. The values can range between 00 and FF (0 and 255, base 10) for each. The format is 0xRRGGBB; either upper or lower case characters are allowed for RR, GG, and BB values. The `0x'' prefix shall have a lower case` x''. The default value is 0xFFFFFF (corresponding to the color white) if this parameter is absent from the request.

When the Format parameter is an image format, a 3DPS **shall** set the background pixels to the color specified by the BackgroundColor parameter.

A 3DPS service **shall** use the BackgroundColor only a) when no Background parameter is provided with the GetView request or b) in the case of an exception and exception format is ``inimage''.

10.4.2. TransparentBackground

The optional TransparentBackground parameter is a flag that specifies whether the view background of a COLOR layer is to be made transparent or not. Default value is ``false''.

NOTE The image/gif format provides 1-bit transparency and is properly displayed by common web clients. The image/png format provides a range of transparency options but support in viewing applications is often implemented slightly at odds with the specification (which is precise regarding alpha blending). The image/jpeg format does not provide transparency.

When TransparentBackground is set to true and the Format parameter contains an image format (e.g., image/gif), then a 3DPS **shall** return (when permitted by the requested format) a result where all of the pixels not representing features or data values in that Layer are set to a transparent value. If the picture format does not support transparency, then the service **shall** respond with a non-transparent image (in other words, it is not an error for the client to always request transparent maps regardless of format). When the TransparentBackground parameter is set to ``false'', non-data pixels **shall** be set to the value of BackgroundColor.

A 3DPS service **shall** generate a transparent background only, when no Background and BackgroundColor parameters are provided with the GetView request.

10.4.3. Portrayals

The mandatory Portrayals parameter specifies one or multiple portrayals to be generated by a 3DPS. It contains one or multiple Portrayal components, which specify image size (width and height), projection, image layers, output formats, and image qualities as defined in Table 46.

Table 46. Components of view:Portrayal structure

Names	Definition	Data type and value	Multiplicity and use
width Width	Width of desired output image, in pixels	PositiveInteger type	One (mandatory)
height Height	Height of desired output image, in pixels	PositiveInteger type	One (mandatory)
projection Projection	Camera specification and projection parameters	core:ProjectionBase type, not empty, see Table 39 and Table 40	One (mandatory)
imageLayers ImageLayers	Reference(s) to image layers to retrieve	List of string elements, not empty	One (mandatory)
formats Formats	Format encoding(s) of ImageLayer(s)	List of ows:MimeType, not empty, see [OWS Common] Subclause 10.5. [a]	One (mandatory)
qualities Qualities	Integer that specifies desired quality of portrayal view (e.g., data resolution, rendering accuracy)	List of Integer type, can be empty List items can be empty, values from 0 to 100, default is 100 [b,c]	Zero or one (optional) Include when desired

[a] The Formats list must have same length as in parameter ImageLayers.
[b] If not empty, the Qualities list must have same length as lists in parameters ImageLayers and Formats.
[c] The Qualities parameter is only useful for formats supporting lossy compression. For other output formats the Qualities parameter shall be ignored. The Qualities parameter should be used carefully: Applying lossy image compression to image layers other than COLOR will result in images containing errors, e.g., wrong depth values or object identifiers.

Width, Height

The mandatory Width and Height parameters specify the size in integer pixels of the 3D view that **shall** be generated. The image CS (see Image coordinate system) applies to the image. *Width-1* specifies the maximum value of the x-axis in the image CS, and *Height-1* specifies the maximum value of the y-axis in the image CS.

If the request is for an image format, the returned picture, regardless of its MIME type, **shall** have exactly the specified width and height in pixels.

Projection

The mandatory Projection parameter specifies a projection to apply for generating the 3D view(s). A projection contains the specification of the virtual camera and projection parameters as defined in 3D projections, Table 39 and Table 40.

ImageLayers

The ImageLayers parameter contains a list of references to one or more image layers as advertised by the 3DPS in it's service metadata, e.g., one of those listed in Table 35.

Formats

The mandatory Formats parameter specifies a list of image formats as ows:MimeType, see Subclause 10.5 of [OWS Common]. The length of this list **shall** be equal to the length of the list in the ImageLayers parameter. The order of the list **shall** correlate with the list in the ImageLayers parameter, meaning that format *n* **shall** be applied for encoding the data of image layer *n*. Each entry in this list **shall** refer to a MIME type as described in the service's metadata (see Table 42).

Qualities

The optional Qualities parameter specifies a list of integers, which describe the visual quality of the output for each requested image layer and format . Values are between 0 and 100. Empty values are possible and refer to the default value (100). If the list is empty, the default value is applied for all of the requested image layers.

The Qualities parameter is mainly useful for COLOR images; for other image layer types, the compression algorithms can generate invalid data. Additionally, the Qualities parameter is only useful for output formats that support image compression. If a requested output format does not support different qualities, the 3DPS **shall** ignore the parameter for this format.

10.4.4. Exceptions

The optional Exceptions parameter defined by core:AbstractGetPortrayal states the format in which to report errors during processing a GetView request. In addition to the parameter values defined by core:AbstractGetPortrayal, this *View* Extension adds the special value ``inimage'' to the list of possible parameter values.

If the Exceptions parameter is set to ``inimage'', the service **shall**, upon detecting an error, return an object of the MIME type specified in the Formats parameter whose content includes text describing the nature of the error. In the case of a picture format, the error message **shall** be drawn on the returned picture.

10.5. GetView response

The normal response to a valid GetView operation request **shall** be

a. if a single image layer is requested: a document of the MIME type as specified in the Formats parameter of the GetView request,

b. if multiple 3D views or image layers are requested: a multipart/mixed message containing image layers of the MIME type as specified in the Formats parameter for each image layer.

10.5.1. MIME multipart response

If multiple portrayals and/or image layers are requested, they **shall** be responded as MIME multipart/mixed content in an HTTP response according to Section 5.1.3 of [RFC 2046]. Each responded image layer **shall** be encoded as one part of that message as follows.

The parts of this multipart/mixed message are separated by the string ``3DPS_MULTIPART_MESSAGE_BOUNDARY'', which is also indicated in the Content-Type header of the multipart response message. According to [IETF RFC 2046], a multipart message also ends with the message's boundary string.

Each part of the multipart response **shall** be a document of a MIME type as specified for this image layer in the request's Formats parameter. Each part contains a header that declares the MIME type of this message part.

If multiple Portrayal elements and/or ImageLayer elements are requested, the resulting image layers in the GetView multipart response **shall** have the same order as in the GetView request.

In the case that not all requested image layers can be generated, a 3DPS service **shall** not respond any image layer, but **shall** respond with an appropriate exception message.

A 3DPS client that requests multiple image layers needs to be capable to parse the multipart response and to extract the message parts.

10.6. GetView exceptions

A GetView request may return any of the exception reports defined in core:AbstractGetPortrayal and/or as specified in Clause 8 of [OWS Common].

Requirement 16: http://www.opengis.net/spec/3DPS/1.0/req/service/view/exception/common
A GetView request, upon encountering an error, **shall** return any of the exception reports defined in core:AbstractGetPortrayal and/or as specified in Clause 8 of [OWS Common].
Dependency: 3DPS Core (http://www.opengis.net/spec/3DPS/1.0/conf-class/core), OWS Common 2.0

10.7. Binding Extensions for the GetView operation

10.7.1. HTTP/GET + KVP binding

This clause specifies the KVP encoding for the GetView operation.

EXAMPLE: A possible GetView request might look like this:
http://hostname:port/path?SERVICE=3DPS&VERSION=1.0

&REQUEST=GetView&CRS=EPSG:26916&BOUNDINGBOX=202759.0,3310170.0,30.0,213200.0,3320896.0,100.0
&SPATIALSELECTION=contains_center&LAYERS=all&STYLES=&OVERALLSTYLES=foggy,abstract
&BACKGROUND=skybox&PORTRAYALS=WIDTH=1024;HEIGHT=1024;
&Projection=Perspective,210000,332000,50,211000,332000,30,0,0,1,60,,0.01,1000.0
&IMAGELAYERS=COLOR;FORMATS=image/jpeg;QUALITIES=85&EXCEPTIONS=INIMAGE

Servers may implement HTTP/GET transfer of the GetView operation request, using KVP encoding. The KVP encoding of the GetView operation request **shall** use the parameters specified in Table 47. The parameters listed in Table 47 **shall** be as specified in Table 45 and Table 46.

Table 47. view:GetView operation request URL parameters

Name[a]	**Optionality and use**	**Definition and format**	**Example**
service	Mandatory	Service type identifier	service=3DPS
request	Mandatory	Operation name	request=GetView
version	Mandatory	Standard and schema version for this operation	version=1.0
crs	Mandatory	CRS to apply to BoundingBox and Viewpoint(s)	crs=EPSG:26916
boundingBox	Optional, include when spatial selection by bounding box desired	Bounding box surrounding desired subset of layer(s), in desired CRS	boundingbox=202759.0,3310170.0,30.0, 213200.0,3320896.0,100.0
spatialSelection	Optional, include when spatial selection desired	Indicates method of selecting objects with BoundingBox	spatialselection=contains_center

Name[a]	Optionality and use	Definition and format	Example
layers	Mandatory	Identifier(s) of desired data layer(s)	layers=dem,bldgs
styles	Optional, include when named layer style desired	Identifier(s) of desired layer style(s)	styles=default,default
overallStyles	Optional, include when image style desired	Identifier(s) of desired overall image style(s)	overallstyles=foggy,abstract
background	Optional, include when background desired	Identifier of desired background	background=skybox
backgroundColor	Optional, include when background color desired	Background color desired	backgroundcolor=0xFF0000
transparentBackground	Optional, include when transparent background desired	Non-data parts shall be transparent	transparentBackground=true

Name[a]	Optionality and use	Definition and format	Example
portrayals	Mandatory	List of specifications of Width (one), Height (one), Projection (one), ImageLayers (one or more), Formats (one or more), Qualities (one or more)	portrayals= WIDTH=1024;HEIGHT=1024; PROJECTION= Perspective, 202000,3310000,200, 202000,3305000,200, 0,0,1, 60,, 0.01,1000.0; IMAGELAYERS=COLOR,DEPTH; FORMATS=image/jpeg,image/png%3Bmode=32bit; QUALITIES=100,100 @ WIDTH=768;HEIGHT=768; PROJECTION= Perspective, 202000,3310000,200, 202000,3305000,200, 0,0,1, 60,, 0.01,1000.0; IMAGELAYERS=COLOR,DEPTH; FORMATS=image/png,image/png; QUALITIES=100,100 [b]
exceptions	Optional, include when default XML not desired	Format of exceptions	exceptions=INIMAGE

[a] All parameter names listed here are using mostly lower case letters. However, any parameter name capitalization shall be allowed in KVP encoding, see Subclause 11.5.2 of [OWS Common].
[b] The value for this field shall be encoded as specified in KVP encoding of the Portrayals parameter

BoundingBox

The value of the optional BoundingBox parameter is a list of comma-separated real numbers as described in Subclause 10.2.3 of [OWS Common]. The units, ordering, and direction of increment of the X and Y axes are as defined by the request's CRS parameter.

Layers

The mandatory Layers parameter **shall** be a comma-separated list of Layer identifiers as advertised in the service metadata's Contents section.

Styles

The Styles parameter **shall** be a comma-separated list of Style identifiers as advertised in the service's metadata Contents section.

KVP encoding of the Portrayals parameter

The Portrayals parameter lists one or multiple items of Portrayal type (see Table 46) and **shall** be encoded in conformance to the following grammar (using EBNF, Extended Backus-Naur Form notation [ISO/IEC 14977]):

```
Portrayals = Portrayal, {"@", Portrayal};
Portrayal = "WIDTH=", 'image width',
    ";HEIGHT=", 'image height',
    ";PROJECTION=", Projection,
    ";IMAGELAYERS=", ImageLayerList,
    ";FORMATS=", FormatList,
    [";QUALITIES=" QualityList];

Projection = ProjectionIdentifier, {",", ProjectionParameterValue};
ProjectionIdentifier =  "Perspective" | "Orthographic" |
    ? name of other supported projection ?;
ProjectionParameterValue = empty | ? URL encoded value ?;

ImageLayerList = ImageLayer, {",", ImageLayer};
ImageLayer = "COLOR" | "DEPTH" | "OBJECTID" | "NORMAL" | "MASK" |
    ? service-specific image layer identifier ?;

FormatList = Format, {",", Format};
Format = ? URL encoded MIME type ?;

QualityList = (Quality | empty), {",", (Qualitiy | empty)};
Quality = ? quality value ?;
```

KVP encoding of the Portrayal components

The Portrayal components are encoded as follows:

a. An ``at'' symbol (@) **shall** be used to separate one Portrayal value from the next.

b. A semicolon (;) **shall** be used to separate one Portrayal parameter from another.

c. An equal sign (=) **shall** be used to separate a Portrayal parameter name from its value.

d. All Portrayal parameter values **shall** be encoded using the standard Internet practice for encoding URLs [RFC 1738].

URL encoding according to [RFC 1738] is required for parameter values.

EXAMPLE: The encoding of the parameterized MIME type `image/png;mode=32Bit''` is image%2Fpng%3Bmode=32Bit''.

KVP encoding of the Projection parameter

The Projection parameter (which is a component of a Portrayal) **shall** be encoded as comma separated list of tuple values. The size of this tuple **shall** be the same as the number of the parameters of this projection plus one (the preceding Projection Identifier). The order of the projection's parameter values **shall** be the same as in the service metadata. For optional projection parameters (see Table 39 and Table 40) the values can be empty. In this case, the service has to use default values, ignore parameters, or compute values.

10.7.2. GetView request XML encoding (optional)

A 3DPS services may implement HTTP/POST transfer of the GetView operation request using XML encoding. In Annex B: XML Schemas (normative) specifies the contents and structure of a GetView operation request encoded in XML.

EXAMPLE: An example GetView operation request XML-encoded for HTTP/POST is:

```
<?xml version="1.0" encoding="UTF-8"?>
<view:GetView
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:core="http://www.opengis.net/3dps/1.0/core"
  xmlns:view="http://www.opengis.net/3dps/1.0/view"
  xsi:schemaLocation="http://www.opengis.net/3dps/1.0/view ../viewGetView.xsd"
  xmlns:ows="http://www.opengis.net/ows/1.1"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  service="3DPS" request="GetView" version="1.0">
  <core:CRS>EPSG:1234</core:CRS>
  <ows:BoundingBox>
    <ows:LowerCorner>0.0 0.0 0.0</ows:LowerCorner>
    <ows:UpperCorner>100.0 100.0 100.0</ows:UpperCorner>
  </ows:BoundingBox>
  <core:SpatialSelection>contains_center</core:SpatialSelection>
  <core:Layers>
    <ows:Identifier>dem</ows:Identifier>
    <ows:Identifier>bldgs</ows:Identifier>
  </core:Layers>
  <core:Styles>
    <ows:Identifier>nice</ows:Identifier>
    <ows:Identifier>default</ows:Identifier>
  </core:Styles>
  <core:OverallStyles>
    <ows:Identifier>fog</ows:Identifier>
    <ows:Identifier>abstract</ows:Identifier>
  </core:OverallStyles>
  <view:BackgroundColor>0xFF0000</view:BackgroundColor>
  <view:Exceptions>INIMAGE</view:Exceptions>
  <view:Portrayals>
    <view:Portrayal>
      <view:Width>1024</view:Width>
      <view:Height>1024</view:Height>
      <view:PerspectiveProjection>
        <view:POC>100 100 100</view:POC>
```

```
        <view:POI>101 101 101</view:POI>
        <view:Up>0 0 1</view:Up>
        <view:FOVX>80</view:FOVX>
        <view:FOVY>60</view:FOVY>
        <view:NearPlane>0.0001</view:NearPlane>
        <view:FarPlane>10000.0</view:FarPlane>
      </view:PerspectiveProjection>
      <view:ImageLayers>
        <ows:Identifier>COLOR</ows:Identifier>
        <ows:Identifier>DEPTH</ows:Identifier>
        <ows:Identifier>OBJECTID</ows:Identifier>
      </view:ImageLayers>
      <view:Formats>
        <view:Format>image/jpeg</view:Format>
        <view:Format>image/png; mode=32bit</view:Format>
        <view:Format>text/xml</view:Format>
      </view:Formats>
      <view:Qualities>80,,</view:Qualities>
    </view:Portrayal>
    <view:Portrayal>
      <view:Width>1024</view:Width>
      <view:Height>1024</view:Height>
      <view:OrthographicProjection>
        <view:POC>100 100 100</view:POC>
        <view:POI>101 101 101</view:POI>
        <view:Up>0 0 1</view:Up>
        <view:Left>-100</view:Left>
        <view:Right>-100</view:Right>
        <view:Bottom>-100</view:Bottom>
        <view:Top>100</view:Top>
        <view:NearPlane>0.0001</view:NearPlane>
        <view:FarPlane>10000.0</view:FarPlane>
      </view:OrthographicProjection>
      <view:ImageLayers>
        <ows:Identifier>COLOR</ows:Identifier>
        <ows:Identifier>DEPTH</ows:Identifier>
        <ows:Identifier>OBJECTID</ows:Identifier>
      </view:ImageLayers>
      <view:Formats>
        <view:Format>image/jpeg</view:Format>
        <view:Format>image/png; mode=32bit</view:Format>
        <view:Format>text/xml</view:Format>
      </view:Formats>
      <view:Qualities>80,,</view:Qualities>
    </view:Portrayal>
  </view:Portrayals>
</view:GetView>
```

Chapter 11. Info Extension

11.1. Introduction

The *Info* Extension specifies capabilities that allow a client to request information about features within a portrayal from a 3DPS service. The canonical use case for the *Info* Extension is that a user explores the response, e.g., of a GetView or GetScene request, and points at an object within the portrayal for which to obtain more information.

For this, the *Info* Extension specifies three specialized operations that are based on an abstract GetFeatureInfo operation and that implement three different methods to identify the selected feature:

- GetFeatureInfoByRay
- GetFeatureInfoByPosition
- GetFeatureInfoByObjectId

The actual method of how the 3DPS decides which features are selected and what kind of information is returned is left to the 3D Portrayal Service implementation. The GetFeatureInfo operation response can be restricted to a feature identifier with the parameter IdOnly. The feature identifier may be used to access a Web Feature Service (WFS) for further information pertaining that feature. The GetFeatureInfo operation is only supported for those Layers for which the attribute queryable has been defined or inherited as ``true''. A client should not issue a GetFeatureInfo request for other layers. A 3D Portrayal Service **shall** respond with a properly formatted OperationNotSupported exception if it receives a GetFeatureInfo request it does not support. Also, a list of layers **shall** be provided with the GetFeatureInfo request so that the search for attribute information can be restricted to selected datasets.

11.2. Modifications to service capabilities

11.2.1. Modifications to ServiceIdentification

A service announces support of the *Info* Extension to a client by adding the URL identifying this extension to the list of supported extensions delivered in the Capabilities document.

Requirement 17: http://www.opengis.net/spec/3DPS/1.0/req/service/info/extension-identifier
A 3DPS service implementing conformance class *info* of this *Info* Extension **shall** include the following URI in a Profile element of the ServiceIdentification section in a GetCapabilities response: http://www.opengis.net/spec/3DPS/1.0/extension/info/1.0
Dependency: 3DPS Core (http://www.opengis.net/spec/3DPS/1.0/conf-class/core) and one of Scene (http://www.opengis.net/spec/3DPS/1.0/conf-class/scene) or View (http://www.opengis.net/spec/3DPS/1.0/conf-class/view)

11.2.2. Modifications to OperationsMetadata

Requirement 18: http://www.opengis.net/spec/3DPS/1.0/req/service/info/operations-metadata-getfeatureinfo
A 3DPS service implementing conformance class *info* of this *Info* Extension **shall** include an ows:Operation element in the OperationsMetadata section in a GetCapabilities response for each of the implemented GetFeatureInfo operations and set its name attribute to a value as defined in Table 48.
Dependency: 3DPS Core (http://www.opengis.net/spec/3DPS/1.0/conf-class/core)

Table 48. Possible values of name attributes of OperationsMetadata section elements

Value of Operation attribute name	**Meaning of attribute value**
GetFeatureInfoByRay	The GetFeatureInfoByRay operation is implemented by this service
GetFeatureInfoByPosition	The GetFeatureInfoByPosition operation is implemented by this service
GetFeatureInfoByObjectId	The GetFeatureInfoByObjectId operation is implemented by this service

Requirement 19: http://www.opengis.net/spec/3DPS/1.0/req/service/info/operations-metadata-getfeatureinfo-formats
A 3DPS service implementing conformance class *info* of this *Info* Extension **shall** include for each implemented GetFeatureInfo operation a list of output formats offered for that operation as ows:MimeType, advertised as ows:Value elements of an ows:Parameter element of the ows:Operation element of this operation, see Subclause 10.5 of [OWS Common].
Dependency: 3DPS Core (http://www.opengis.net/spec/3DPS/1.0/conf-class/core)

EXAMPLE: A partial example of an OperationsMetadata element is as follows:

```
<OperationsMetadata>
  <Operation name="GetCapabilities">
    <DCP>
      <HTTP>
        <Get xlink:href="http://example.com/transform?"/>
      </HTTP>
    </DCP>
    <Parameter name="Format">
      <Value>text/xml</Value>
    </Parameter>
  </Operation>
  <Operation name="GetFeatureInfoByRay">
    <DCP>
      <HTTP>
        <Get xlink:href="http://example.com/transform?"/>
        <Post xlink:href="http://example.com/transform?"/>
      </HTTP>
    </DCP>
    <Parameter name="Format">
      <Value>text/xml</Value>
      <Value>text/plain</Value>
      <Value>text/html</Value>
    </Parameter>
    <Parameter name="Exceptions">
            <Value>text/xml</Value>
            <Value>text/plain</Value>
            <Value>text/html</Value>
    </Parameter>
  </Operation>
</OperationsMetadata>
```

11.2.3. Additions to Layer structure

The *Info* Extension provides an info:InfoLayerExtension component to add a queryable attribute to the core:Layer structure.

Requirement 20: http://www.opengis.net/spec/3DPS/1.0/req/service/info/layer-extension
A 3DPS service implementing conformance class *info* of this *Info* Extension **shall** extend the core:Layer Extensions structure by zero or one InfoLayerExtension structure as defined in Figure 18 and Table 49
Dependency: 3DPS Core (http://www.opengis.net/spec/3DPS/1.0/conf-class/core)

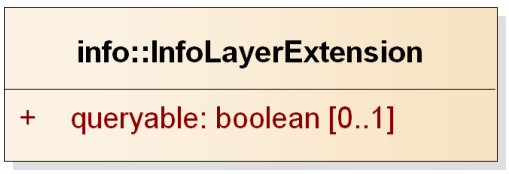

Figure 18. 3DPS info:InfoLayerExtension UML class diagram

Table 49. Components of info:InfoLayerExtension structure

Names	Definition	Data type and value	Multiplicity and use
queryable Queryable	Flag indicating if feature info can be requested from this layer	Boolean type `true'' means the layer is queryable, false'' means layer is not queryable, default is ``false''	Zero or one (optional)

11.3. AbstractGetFeatureInfo request

Requirement 21: http://www.opengis.net/spec/3DPS/1.0/req/service/info/abstractGetFeatureInfo

An info:AbstractGetFeatureInfo request **shall** consist of an info:AbstractGetFeatureInfo structure as defined in Figure 19 and Table 50.

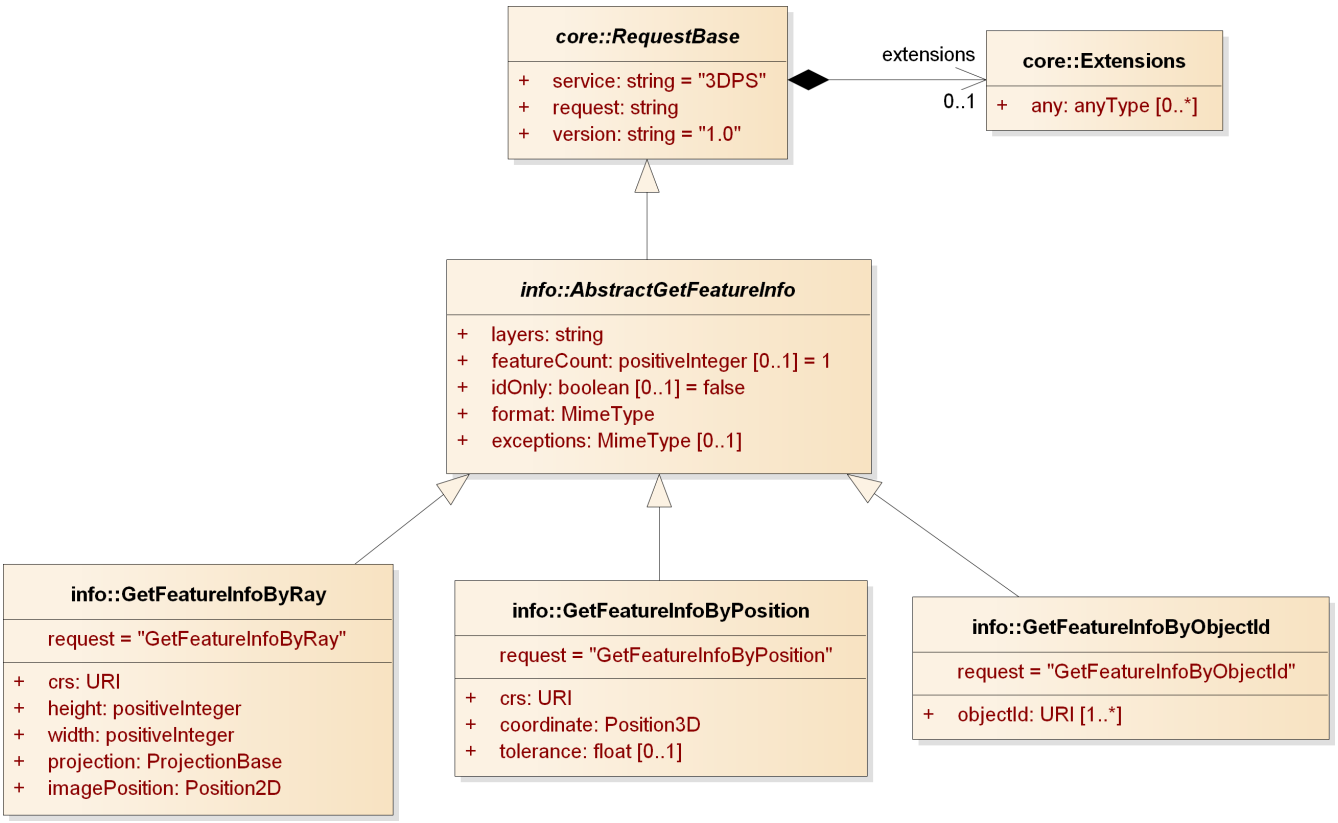

Figure 19. 3DPS info:AbstractGetFeatureInfo operation request UML class diagram

Table 50. Additional components of info:AbstractGetFeatureInfo request

Names	Definition	Data type and value	Multiplicity and use
request request	Operation name	string, not empty Value is one of `GetFeatureInfoByRay'', GetFeatureInfoByPosition'', ``GetFeatureInfoByObjectId''	One (mandatory)

Names	Definition	Data type and value	Multiplicity and use
layers Layers	List of layers to retrieve the data from	StringList	One (mandatory)
featureCount FeatureCount	Number of features to return information from.	PositiveInteger type Default is ``1''	Zero or one (optional)
idOnly IdOnly	restrict feature information to feature id	Boolean type Default is ``false''	Zero or One (optional)
format Format	Format encoding of the result	ows:MimeType, see [OWS Common] Subclause 10.5	One (mandatory)
exceptions Exceptions	Format in which operation exceptions are returned	ows:MimeType, see [OWS Common] Subclause 10.5	Zero or one (optional)

11.3.1. Layers

The Layers parameter specifies a list of layer identifiers from which the service spatially selects features based on the coordinate, and collects attribute information from. The order in which the layers are listed in the Layers parameter determines the order in which the feature information will be displayed in the GetFeatureInfo response. Each entry in the Layers list **shall** refer to a layer identifier as described in the service metadata.

11.3.2. FeatureCount

The optional FeatureCount parameter specifies the maximum number of features per layer for which feature information **shall** be returned. The parameter value **shall** be a positive integer. The default value is ``1'' if this parameter is omitted or is other than a positive integer.

11.3.3. IdOnly

The optional IdOnly parameter can be used to restrict the GetFeatureInfo response to the unique feature identifier only (value is `true''). The feature identifier can be used, e.g., to request further information of the feature from a Web Feature Server. Default value is` false''.

11.3.4. Format

The mandatory Format parameter specifies the target encoding of the returned attribute information provided as ows:MimeType, see Subclause 10.5 of [OWS Common]. Available formats are described in the service metadata.

11.3.5. Exceptions

The optional Exceptions parameter specifies the behavior of the service upon detecting an error, e.g., invalid request or internal service error. The default value is ``text/xml''. The value **shall** be one of the MIME types offered in the service metadata parameter value.

11.4. GetFeatureInfoByRay request

GetFeatureInfoByRay allows a client to request information about features by selecting them through a virtual ray, set up from the virtual camera to a 3D point or 2D point in a 3D scene or 3D view, respectively.

It is assumed that, when performing the ray cast, the scene may not be accurately reproduced inside the service instance for the benefit of item selection. Thus, the request does not feature the additional options of the *View* or *Scene* profile to aid in reproduction. Rather, some amount of error should be assumed when using this request.

The GetFeatureInfoByRay request is derived from the AbstractGetFeatureInfo request and inherits all its parameters. Additional parameters are defined for setting up an intersection ray.

Requirement 22: http://www.opengis.net/spec/3DPS/1.0/req/service/info/getfeatureinfobyray/request An info:GetFeatureInfoByRay request **shall** consist of an info:GetFeatureInfoByRay structure as defined in Figure 19, Table 50, and Table 51.

Table 51. Additional components of info:GetFeatureInfoByRay request

Names	Definition	Data type and value	Multiplicity and use
request Request	Operation name	string Value is fixed to ``GetFeatureInfoByRay''	One (mandatory)
crs CRS	CRS to apply to BoundingBox and Projection parameters	anyURI	One (mandatory)
width Width	Width of output image that the request refers to, in pixels	PositiveInteger type, positive value	One (mandatory)
height Height	Height of output image that the request refers to, in pixels	PositiveInteger type	One (mandatory)
projection Projection	Camera specification and projection parameters	ProjectionBase type Supported projection types are specified in service metadata	One (mandatory)
imagePosition ImagePosition	Discrete pixel position for which to search for features to return information from	Position2D type in image CS, see Table 6	One (mandatory)

11.4.1. Width, Height

The Width and Height parameters specify the size in integer pixels of the view that describes the ray cast.

11.4.2. Projection

The Projection parameter specifies a projection to apply to the ray cast. A projection contains the specification of the virtual camera and projection parameters as defined in 3D projections, Table 39

and Table 40.

11.4.3. ImagePosition

The ImagePosition parameter specifies the 2D pixel position for which to analyze the scene and for where to retrieve information for. The parameter value is a list of two integer numbers describing the X, Y coordinates of the pixel in image CS.

11.5. GetFeatureInfoByPosition request

The GetFeatureInfoByPosition operation finds features based on a location, usually determined in the portrayal by clicking on, or close to, an object. The service locates the closest FeatureCount features and returns the associated feature attributes to the client.

The GetFeatureInfoByPosition request is derived from the AbstractGetFeatureInfo request and inherits all its parameters. Additional parameters are defined to support locating nearby features.

Requirement 23: http://www.opengis.net/spec/3DPS/1.0/req/service/info/getfeatureinfobyposition/request
An info:GetFeatureInfoByPosition request **shall** consist of an info:GetFeatureInfoByPosition structure as defined in Figure 19, Table 50, and Table 52.

Table 52. Additional components of info:GetFeatureInfoByPosition request

Names[a]	**Definition**	**Data type and value**	**Multiplicity and use**
request Request	Operation name	string Value is fixed to ``GetFeatureInfoByPosition''	One (mandatory)
crs CRS	CRS of the coordinates	anyURI	One (mandatory)
coordinate Coordinate	Location coordinates used to search for features	Position3D type, see Table 6, coordinate order in CRS space	One (mandatory)
tolerance Tolerance	Tolerance in meter	A floating-point number	Zero or one (optional)

11.5.1. CRS

The CRS parameter's value is defined in Subclause 10.3 of [OWS Common] and [OGC 04-046r3]. When using a 2D CRS, then the height reference is taken from the layer's vertical datum.

11.5.2. Coordinate

The Coordinate parameter specifies a geolocation within the scene from which feature information will be retrieved. The parameter value is a list of comma separated floating point numbers describing a geoposition in CRS coordinates. This location should be within or at the border of a

feature geometry for accuracy reasons, but it does not need to. The 3D Portrayal Service **shall** detect the feature(s) whose geometry is covering the location or which are close to the location.

11.5.3. Tolerance

Optionally, a tolerance in meter may be specified which guides the service in determining the range in which features should be located. It is provisional only, i.e., the service may return features outside that range even when specified, and it also may use an implementation-defined default if not specified.

11.6. GetFeatureInfoByObjectId request

Some (mostly 3D) formats allow for inclusion of object identifiers. Such identifiers may then be used to obtain feature attributes directly. Except through the specified IdOnly mechanism, the details of how precisely object identifiers are obtained in the client are left unspecified. When an object identifier is obtained and further information is sought, the GetFeatureInfoByObjectId operation can be used directly to request additional information.

The GetFeatureInfoByObjectId request is derived from the AbstractGetFeatureInfo request and inherits all its parameters. Additional and refined parameters are defined to specify a list of desired identifiers.

Requirement 24: http://www.opengis.net/spec/3DPS/1.0/req/service/info/getfeatureinfobyobjectid/request
An info:GetFeatureInfoByObjectId request **shall** consist of an info:GetFeatureInfoByRay structure as defined in Figure 19, Table 50, and Table 53.

Table 53. Additional components of info:GetFeatureInfoByObjectId request

Names	Definition	Data type and value	Multiplicity and use
request Request	Operation name	string Value is fixed to ``GetFeatureInfoByPosition''	One (mandatory)
objectId ObjectId	Identifier of the selected feature(s)	URI	One or more (mandatory)
featureCount FeatureCount	Number of features to return information from.	Integer type, default is the number of ObjectId identifiers provided.	Zero or one (optional)

11.6.1. ObjectID

Unique identifier(s) of the selected feature(s).

11.6.2. FeatureCount

If FeatureCount is specified, it **shall** correspond to the number of object identifiers provided.

Otherwise, FeatureCount **shall** be assumed to equal the cardinality of the ObjectId identifiers.

11.7. GetFeatureInfo response

The normal response to a valid GetFeatureInfo operation request **shall** be a document in which the attribute information of each identified feature is listed. The response is a document encoded in the MIME type as specified in the Format parameter.

Requirement 25: http://www.opengis.net/spec/3DPS/1.0/req/service/info/response
To a valid GetFeatureInfo request, a 3DPS service implementing conformance class *info* of this *Info* Extension **shall** respond a FeatureInfo as defined in Table 54, Table 55, Table 56 and Table 57.
Dependency: 3DPS Core (http://www.opengis.net/spec/3DPS/1.0/conf-class/core)

NOTE	A FeatureInfoList can contain multiple more than one FeatureAttributeList elements, which allows, e.g., for grouping feature attributes by their topic or domain.

Table 54. Components of info:FeatureInfo structure

Names	**Definition**	**Data type and value**	**Multiplicity and use**
featureInfoList FeatureInfoList	List containing information of one feature	FeatureInfoList type	One or more (mandatory) Include one for each feature that was found

Table 55. Components of info:FeatureInfoList structure

Names	**Definition**	**Data type and value**	**Multiplicity and use**
featureId FeatureId	Feature identifier that may be used to directly request an underlying feature database (e.g., OGC WFS)	string, not empty	Zero or one (optional) Include if feature identifier is available
objectId ObjectId	Object identifier, as used by this server [a]	string, not empty	Zero or one (optional) Include if object identifier is available
typeName TypeName	Name of the feature's type	string, not empty	Zero or one (optional) Include if a type name is available

Names	Definition	Data type and value	Multiplicity and use
featureAttributeList FeatureAttributeList	Group of feature attributes	FeatureAttributeList data type, not empty, see Table 56	Zero or more (optional) Include when attributes available for this feature
[a] The object identifiers can be assumed valid only for this server. In that sense, it differes from a feature identifier, which should be unique across multiple servers' data stores.			

Table 56. Components of info:FeatureAttributeList structure

Names	Definition	Data type and value	Multiplicity and use
attribute Attribute	Feature attribute name and value	FeatureAttribute type, not empty, see Table 57	One or more (mandatory)

Table 57. Components of info:FeatureAttribute structure

Names	Definition	Data type and value	Multiplicity and use
name Name	Attribute name	string, not empty	One (mandatory)
value Value	Attribute value	string, might be empty	One (mandatory)

11.7.1. FeatureInfo encoding

FeatureInfo XML encoding

Example: An example XML-encoded FeatureInfo response is:

```
<FeatureInfo xmlns="http://www.opengis.net/3dps/1.0/info">
  <FeatureInfoList featureType="buildings">
    <ObjectId>496376</ObjectId>
    <FeatureId>BLDG_0003000e0080fce9</FeatureId>
    <FeatureAttributeList>
      <FeatureAttribute name="id">34891270</FeatureAttribute>
      <FeatureAttribute name="objkey">1123 Hochschulgebaeude</FeatureAttribute>
      <FeatureAttribute name="strkey">2390</FeatureAttribute>
      <FeatureAttribute name="hsno">1</FeatureAttribute>
      <FeatureAttribute name="description">Alte Universitaet</FeatureAttribute>
    </FeatureAttributeList>
    <FeatureAttributeList>
      <FeatureAttribute name="sockelhoehe">2.500000</FeatureAttribute>
      <FeatureAttribute name="traufhoehe">15.000000</FeatureAttribute>
      <FeatureAttribute name="height">115.347270</FeatureAttribute>
    </FeatureAttributeList>
  </FeatureInfoList>
</FeatureInfo>
```

FeatureInfo HTML encoding

A GetFeatureInfo operation response may look like this encoded in text/html:

```
<html>
  <head></head>
  <body>
    <table>
      <tr>
        <th><b>Buildings</b></th>
      </tr>
      <tr>
        <th></th>
        <th>id</th>
        <th>objkey</th>
        <th>strkey</th>
        <th>hsno</th>
        <th>description</th>
        <th>sockelhoehe</th>
        <th>traufhoehe</th>
        <th>height</th>
      </tr>
      <tr>
        <td>1</td>
        <td>34891270</td>
        <td>1123 Hochschulgebaeude</td>
        <td>2390</td>
        <td>1</td>
        <td>Alte Universitaet</td>
        <td>2.500000</td>
        <td>15.000000</td>
        <td>115.347270</td>
      </tr>
    </table>
  </body>
</html>
```

11.7.2. GetFeatureInfo exceptions

When a 3DPS encounters an error while processing a GetFeatureInfo request, it **shall** return an exception report message as specified in Clause 8 of [OWS Common]. The allowed standard exception codes **shall** include those listed in Table 58. For each listed exceptionCode, the contents of the ``locator'' parameter value **shall** be filled as specified in the right column of the table.

11.7.3. Exception codes

Table 58. Exception codes of the GetFeatureInfo operations.

exceptionCode value	**Meaning of code**	**``locator'' value**
OperationNotSupported	Request is for an operation that is not supported by this service	Name of operation not supported

exceptionCode value	Meaning of code	``locator'' value
MissingParameterValue	Operation request does not include a parameter value, and this service did not declare a default value for that parameter	Name of missing parameter
InvalidParameterValue	Operation request contains an invalid parameter value	Name of parameter with invalid value
OptionNotSupported	Request is for an option that is not supported by this service	Identifier of option not supported
CRSNotSupported	Operation request contains a value in the CRS parameter which is not supported by the service	Name of unsupported CRS
UnknownLayer	Operation request contains an identifier in the Layers parameter which is unknown to the service	Identifier of invalid layer
FormatNotSupported	Operation request contains a MIME type in the Format parameter which is not supported by the service	Name of unsupported format
ExceptionNotSupported	Operation request contains a value in the Exception parameter which is not supported by the service	Name of unsupported exception format
NoApplicableCode	No other exceptionCode specified by this service and service applies to this exception	None, omit ``locator'' parameter

Chapter 12. Annex A: Conformance Class Abstract Test Suite (normative)

Tests and requirement identifiers below are relative to http://www.opengis.net/spec/3DPS/1.0/testing.

References to XML schema read either ows:Type or core:Type and refer to http://www.opengis.net/ows/2.0 or http://www.opengis.net/3dps/1.0/core, respectively.

12.1. Conformance class: core

Conformance class: core	
The OGC URI identifier of this conformance class is: http://www.opengis.net/spec/3DPS/1.0/conf-class/core	
Test identifier	http://www.opengis.net/spec/3DPS/1.0/testing/conf-class/core
Test purpose	Verify that the service implements the ``core'' 3DPS conformance class.
Test method	Verify that the service declares and implements the core conformance class by evaluating the GetCapabilities response. Verify that the requests and responses to a supported operation are syntactically correct. Verify that the service supports the view conformance class, the scene conformance class or both.

12.2. Tests for requirements of the core requirements class

Most core requirements are actually requirements against extensions to this standard. They are therefore not included here.

GetCapabilities request	
Requirement 1	
Test identifier	http://www.opengis.net/spec/3DPS/1.0/req/service/core/getcapabilities/request/structure
Test purpose	Test that the server can handle GetCapabilities requests.
Test method	Send a valid GetCapabilities request to the server under test. Test passes if a valid document of the type core:CapabilitiesType is returned.

GetCapabilities response	
Requirement 2	
Test identifier	http://www.opengis.net/spec/3DPS/1.0/req/service/core/getcapabilities/response/structure
Test purpose	Test that the GetCapabilities response conforms to the specification.
Test method	Verify that the GetCapabilities response is valid according to the declared schema and that the root element conforms to core:CapabilitiesType.

GetResourceById request	
Requirement 3	
Test identifier	http://www.opengis.net/spec/3DPS/1.0/req/service/core/getResourceById/request
Test purpose	Test that the GetResourceById response conforms to the specification.
Test method	Verify the service's core:GetResourceById against [OGC 06-121r9] Section A.3.3 with the additional constraints laid out in this section.'

GetResourceById response	
Requirement 4	
Test identifier	http://www.opengis.net/spec/3DPS/1.0/req/service/core/getResourceById/response
Test purpose	Test that the response is formed as expected.
Test method	Request data from the service and validate its inner structure and the service response's MIME type to match the requested MIME type.'

12.3. Conformance class: scene

Conformance class: scene	
The OGC URI identifier of this conformance class is: http://www.opengis.net/spec/3DPS/1.0/conf-class/scene	
Test identifier	http://www.opengis.net/spec/3DPS/1.0/testing/profile/scene
Test purpose	Verify that the service implements the ``scene'' 3DPS conformance class.
Test method	Verify that the service declares and implements the GetScene operation. Verify that GetScene declares at least one format. Verify conformance test {TestPrefix}/conf-class/core.

12.4. Tests for requirements of the scene requirements class

NOTE These tests require an instance that has some content.

Modifications to service identification	
Requirement 5	
Test identifier	http://www.opengis.net/spec/3DPS/1.0/req/service/scene/extension-identifier
Test purpose	Test that an instance declares its implementation of the scene requirements class.
Test method	Verify that the GetCapabilities response includes the declaration of a profile element {SpecElementPrefix}/extension/scene/1.0

Test for Operations Metadata	
Requirement 6	
Test identifier	http://www.opengis.net/spec/3DPS/1.0/req/service/scene/operations-metadata-getscene
Test purpose	Test that operations metadata is complete.
Test method	Verify that the GetCapabilities response includes an operation named `GetScene'. Verify that the operation declares the parameters given in Table 24 and Table 32 are declared.

Layer extension test	
Requirement 7	
Test identifier	http://www.opengis.net/spec/3DPS/1.0/req/service/scene/layer-extension
Test purpose	Ensure that the layer extensions for scene-based portrayal are declared.
Test method	Verify that at least one layer includes a maximum of one scene:SceneLayerExtension element in its Extensions element.

Mime types	
Requirement 8	
Test identifier	http://www.opengis.net/spec/3DPS/1.0/req/service/scene/mimetypes
Test purpose	Test that declared MIME types conform to the specification.
Test method	Verify that there are MIME types available for each layer. Verify that for each parameterized MIME type, there is a corresponding unparameterized MIME.

Portrayal capabilities declaration test	
Requirement 9	
Test identifier	http://www.opengis.net/spec/3DPS/1.0/req/service/scene/portrayalcapabilities-extension
Test purpose	Test that portrayal capabilities are declared properly.
Test method	Verify that the GetCapabilities response contains an extension element scene:ScenePortrayalCapabilitiesExtension.

Response	
Requirement 11	
Test identifier	http://www.opengis.net/spec/3DPS/1.0/req/service/scene/getscene/response
Test purpose	Test that the response is formed as expected.
Test method	Request data from all layers from the service in all declared MIME types, individually. Verify that each response is formed according to the provisions of the format associated with the MIME type used for that response, or is an error. Verify that at least one response is not an error.

12.5. Conformance class: view

Conformance class: view	
The OGC URI identifier of this conformance class is: http://www.opengis.net/spec/3DPS/1.0/conf-class/view	
Test identifier	http://www.opengis.net/spec/3DPS/1.0/testing/profile/view
Test purpose	Verify that the service implements the ``view" 3DPS conformance class.
Test method	Verify that the service declares and implements the GetView operation. Verify that GetView declares at least one format. Verify conformance test {TestPrefix}/conf-class/core.

12.6. Tests for requirements of the view requirements class

Modifications to service identification	
Requirement 12	
Test identifier	http://www.opengis.net/spec/3DPS/1.0/req/service/view/extension-identifier
Test purpose	Test that an instance declares its implementation of the view requirements class.
Test method	Verify that the GetCapabilities response includes the declaration of a profile element {SpecElementPrefix}/extension/view/1.0

Test for Operations Metadata	
Requirement 13	
Test identifier	http://www.opengis.net/spec/3DPS/1.0/req/service/view/operations-metadata-getview
Test purpose	Test that operations metadata is complete.
Test method	Verify that the GetCapabilities response includes an operation named "GetView". Verify that the operation declares the parameters given in Table 24 and Table 45 are declared.

Portrayal capabilities declaration test	
Requirement 14	
Test identifier	http://www.opengis.net/spec/3DPS/1.0/req/service/view/portrayalcapabilities-extension
Test purpose	Test that portrayal capabilities are declared properly.
Test method	Verify that the GetCapabilities response contains an extension element view:ViewPortrayalCapabilitiesExtension.

Test exception use in response	
Requirement 16	
Test identifier	http://www.opengis.net/spec/3DPS/1.0/req/service/view/exception/common

Test exception use in response	
Test purpose	Test that the response is formed as expected.
Test method	Request data from all layers from the service in all declared MIME types, individually. Verify that each response is formed according to the provisions of the format associated with the MIME type used for that response, or is an error. Verify that at least one response is not an error.

12.7. Conformance class: info

Conformance class: info	
The OGC URI identifier of this conformance class is: http://www.opengis.net/spec/3DPS/1.0/conf-class/info	
Test identifier	http://www.opengis.net/spec/3DPS/1.0/testing/profile/info
Test purpose	Verify that the service implements the ``info'' 3DPS conformance class.
Test method	Verify that the service declares and implements one of GetFeatureInfoByRay, GetFeatureInfoByPosition or GetFeatureInfoByObjectID operations. Verify conformance test {TestPrefix}/conf-class/core. Verify either conformance test conf-class/scene or conf-class/view or both.

12.8. Tests for requirements of the info requirements class

Modifications to service identification	
Requirement 17	
Test identifier	http://www.opengis.net/spec/3DPS/1.0/req/service/info/extension-identifier
Test purpose	Test that an instance declares its implementation of the info requirements class.
Test method	Verify that the GetCapabilities response includes the declaration of a profile element {SpecElementPrefix}/extension/info/1.0

Test for Operations Metadata	
Requirement 18	
Test identifier	http://www.opengis.net/spec/3DPS/1.0/req/service/info/operations-metadata-getfeatureinfo
Test purpose	Test that operations metadata contains at least one info-request operation
Test method	Verify that the GetCapabilities response includes at least one operation mentioned in Table 48.

Test for feature information formats	
Requirement 19	
Test identifier	http://www.opengis.net/spec/3DPS/1.0/req/service/info/operations-metadata-getfeatureinfo-formats

Test for feature information formats	
Test purpose	Test that feature information formats are declared
Test method	Verify that each operation from Table 48 in the operations metadata declares a format parameter and one or more possible values.

Test for the layer extension	
Requirement 20	
Test identifier	http://www.opengis.net/spec/3DPS/1.0/req/service/info/layer-extension
Test purpose	Test that the layer extension is used
Test method	Verify that at least one layer includes a maximum of one scene:SceneLayerExtension element in its Extensions element.

Test the abstract feature info generalization	
Requirement 21	
Test identifier	http://www.opengis.net/spec/3DPS/1.0/req/service/info/abstractGetFeatureInfo
Test purpose	Test that all feature info operations extend AbstractGetFeatureInfo
Test method	Verify that each operation from Table 48 declared in the operations metadata also declares the parameters listed in Table 50.

Test GetFeatureInfoByRay request structure	
Requirement 22	
Test identifier	http://www.opengis.net/spec/3DPS/1.0/req/service/info/getfeatureinfobyray/request
Test purpose	Test that the request structure conforms to this standard
Test method	Verify that the request structure as declared is consistent with Figure 19, Table 51 and Table 50. Verify that all parameters are declared.

Test GetFeatureInfoByPosition request structure	
Requirement 23	
Test identifier	http://www.opengis.net/spec/3DPS/1.0/req/service/info/getfeatureinfobyposition/request
Test purpose	Test that the request structure conforms to this standard
Test method	Verify that the request structure as declared is consistent with Figure 19, Table 52 and Table 50. Verify that all parameters are declared.

Test GetFeatureInfoByObjectId request structure	
Requirement 24	
Test identifier	http://www.opengis.net/spec/3DPS/1.0/req/service/info/getfeatureinfobyobjectid/request
Test purpose	Test that the request structure conforms to this standard
Test method	Verify that the request structure as declared is consistent with Figure 19, Table 53 and Table 50. Verify that all parameters are declared.

Test exception use in response	
Requirement 25	
Test identifier	http://www.opengis.net/spec/3DPS/1.0/req/service/info/response
Test purpose	Test that the response is formed as expected.
Test method	Request feature info from all queryable layers from the service in all declared MIME types, individually. Verify that each response is formed according to the provisions of the format associated with the MIME type used for that response, or is an error. Verify that at least one response is not an error.

Chapter 13. Annex B: XML Schemas (normative)

The XML schema documents corresponding to XML 3DPS queries and responses are to be found in http://schemas.opengis.net/3dps/1.0. The schema content is being repeated here as a convenience to the reader. The schemas on http://schemas.opengis.org are considered normative.

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

The XML schemas are being developed jointly with the standard documents, and care has been taken that every well-formed (in the XML sense) example validates against the schema.

The XML schema documents use (i.e. import) and build on the OWS common XML Schema Documents specified in [OWS Common], named:

- ows19115subset.xsd
- owsCommon.xsd
- owsDataIdentification.xsd
- owsExceptionReport.xsd
- owsGetCapabilities.xsd
- owsOperationsMetadata.xsd
- owsServiceIdentification.xsd
- owsServiceProvider.xsd

13.1. core schema

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.opengis.net/3dps/1.0/core">
  <xs:import namespace="http://www.opengis.net/ows/2.0"
schemaLocation="http://schemas.opengis.net/ows/2.0/owsContents.xsd"/>
  <xs:import namespace="http://www.opengis.net/ows/2.0"
schemaLocation="http://schemas.opengis.net/ows/2.0/owsDataIdentification.xsd"/>
  <xs:import namespace="http://www.opengis.net/ows/2.0"
schemaLocation="http://schemas.opengis.net/ows/2.0/owsCommon.xsd"/>
  <xs:complexType name="AbstractGetPortrayalType" abstract="true">
    <xs:complexContent>
      <xs:extension base="RequestBaseType">
        <xs:sequence>
          <xs:element name="CRS" type="xs:anyURI" minOccurs="1" maxOccurs="1"/>
          <xs:element name="BoundingBox" type="ows:BoundingBoxType" minOccurs="1"
maxOccurs="1"/>
          <xs:element name="SpatialSelection" type="xs:string" minOccurs="0"
```

```
maxOccurs="1"/>
          <xs:element name="Layers" type="xs:string" minOccurs="1" maxOccurs="1"/>
          <xs:element name="Styles" type="xs:string" minOccurs="0" maxOccurs="1"/>
          <xs:element name="Background" type="xs:string" minOccurs="0" maxOccurs="1"/>
          <xs:element name="LODs" type="xs:string" minOccurs="0" maxOccurs="1"/>
          <xs:element name="LODSelection" type="xs:string" minOccurs="0" maxOccurs=
"1"/>
          <xs:element name="OverallStyles" type="xs:string" minOccurs="0"
maxOccurs="1"/>
          <xs:element name="DeliveryOptions" type="xs:string" minOccurs="0"
maxOccurs="unbounded"/>
          <xs:element name="Exceptions" type="xs:string" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="CapabilitiesType">
    <xs:complexContent>
      <xs:extension base="ows:CapabilitiesBaseType">
        <xs:sequence>
          <xs:element name="Contents" type="ContentsType" minOccurs="0" maxOccurs=
"1"/>
          <xs:element name="PortrayalCapabilities" type="PortrayalCapabilitiesType"
minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="RequestBaseType" abstract="true">
    <xs:sequence>
      <xs:sequence minOccurs="0" maxOccurs="1" ObjectID="1624"/>
    </xs:sequence>
    <xs:attribute name="service" use="required" type="xs:string"/>
    <xs:attribute name="request" use="required" type="xs:string"/>
    <xs:attribute name="version" use="required" type="xs:string"/>
  </xs:complexType>
  <xs:complexType name="DeliveryOptionType">
    <xs:complexContent>
      <xs:extension base="ows:DescriptionType">
        <xs:sequence>
          <xs:element ref="ows:Identifier" minOccurs="1" maxOccurs="1"/>
          <xs:element name="Format" type="ows:MimeType" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="ViewpointHintType">
    <xs:complexContent>
      <xs:extension base="ows:DescriptionType">
        <xs:sequence>
```

```
          <xs:element name="POC" type="Position3DType" minOccurs="1" maxOccurs="1"/>
          <xs:element name="POI" type="Position3DType" minOccurs="1" maxOccurs="1"/>
          <xs:element name="UP" type="Position3DType" minOccurs="1" maxOccurs="1"/>
          <xs:element name="FOVX" type="xs:double" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:simpleType name="Section">
    <xs:restriction base="xs:string">
      <xs:enumeration value="ServiceIdentification"/>
      <xs:enumeration value="ServiceProvider"/>
      <xs:enumeration value="OperationsMetadata"/>
      <xs:enumeration value="Languages"/>
      <xs:enumeration value="Contents"/>
      <xs:enumeration value="PortrayalCapabilities"/>
      <xs:enumeration value="All"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="PortrayalCapabilitiesType">
    <xs:sequence>
      <xs:element name="AvailableSpatialSelection" type="xs:string" minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element name="AvailableLODSelection" type="xs:string" minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element name="SupportsBoundingBoxConversion" type="xs:boolean" minOccurs="0"
maxOccurs="1"/>
      <xs:element name="AvailableLODScheme" type="LODSchemeType" minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element name="ViewpointHint" type="ViewpointHintType" minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element name="DeliveryOption" type="DeliveryOptionType" minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element name="OverallStyle" type="OverallStyleType" minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element name="Background" type="BackgroundType" minOccurs="0"
maxOccurs="unbounded"/>
      <xs:sequence minOccurs="0" maxOccurs="1" ObjectID="1624"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="LODSchemeType">
    <xs:complexContent>
      <xs:extension base="ows:DescriptionType">
        <xs:sequence>
          <xs:element ref="ows:Identifier" minOccurs="1" maxOccurs="1"/>
          <xs:element name="LOD" type="xs:QName" minOccurs="1" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="LayerType">
```

```
    <xs:complexContent>
      <xs:extension base="ows:DescriptionType">
        <xs:sequence>
          <xs:element ref="ows:Identifier" minOccurs="1" maxOccurs="1"/>
          <xs:element name="AvailableCRS" type="xs:anyURI" minOccurs="1"
maxOccurs="unbounded"/>
          <xs:element name="AvailableLOD" type="xs:string" minOccurs="0"
maxOccurs="unbounded"/>
          <xs:element name="DeliveryOption" type="xs:string" minOccurs="0"
maxOccurs="unbounded"/>
          <xs:element name="Layer" type="LayerType" minOccurs="0"
maxOccurs="unbounded"/>
          <xs:sequence minOccurs="0" maxOccurs="1"/>
          <xs:element name="AvailableStyle" type="StyleType" minOccurs="0"
maxOccurs="unbounded"/>
          <xs:element name="WGS84BoundingBox" type="ows:WGS84BoundingBoxType"
minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="Metadata" type="ows:MetadataType" minOccurs="0"
maxOccurs="unbounded"/>
          <xs:element name="BoundingBox" type="ows:BoundingBoxType" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="GetResourceByIdType">
    <xs:complexContent>
      <xs:extension base="RequestBaseType">
        <xs:sequence>
          <xs:element name="ResourceID" type="xs:anyURI" minOccurs="1" maxOccurs="1"/>
          <xs:element name="OutputFormat" type="xs:string" minOccurs="1" maxOccurs=
"1"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="Position3DType">
    <xs:sequence>
      <xs:element name="X1" type="xs:double" minOccurs="1" maxOccurs="1"/>
      <xs:element name="X2" type="xs:double" minOccurs="1" maxOccurs="1"/>
      <xs:element name="X3" type="xs:double" minOccurs="1" maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="ContentsType">
    <xs:complexContent>
      <xs:extension base="core:OWSContentsBaseRestrictionType">
        <xs:sequence minOccurs="0" maxOccurs="1">
          <xs:element name="Layer" type="LayerType" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
```

```
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="GetCapabilitiesType">
    <xs:complexContent>
      <xs:extension base="ows:GetCapabilitiesType">
        <xs:sequence/>
        <xs:attribute name="service" use="optional" type="xs:string"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="OverallStyleType">
    <xs:complexContent>
      <xs:extension base="ows:DescriptionType">
        <xs:sequence>
          <xs:element ref="ows:Identifier" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="BackgroundType">
    <xs:complexContent>
      <xs:extension base="ows:DescriptionType">
        <xs:sequence>
          <xs:element ref="ows:Identifier" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="StyleType">
    <xs:complexContent>
      <xs:extension base="ows:DescriptionType">
        <xs:sequence>
          <xs:element ref="ows:Identifier" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="Position2DType">
    <xs:sequence>
      <xs:element name="X1" type="xs:double" minOccurs="1" maxOccurs="1"/>
      <xs:element name="X2" type="xs:double" minOccurs="1" maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="OWSContentsBaseRestrictionType">
    <xs:complexContent>
      <xs:extension base="ows:ContentsBaseType">
        <xs:sequence/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:schema>
```

13.2. scene schema

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.opengis.net/3dps/1.0/scene">
  <xs:import namespace="http://www.opengis.net/ows/2.0"
schemaLocation="http://schemas.opengis.net/ows/2.0/ows19115subset.xsd"/>
  <xs:import namespace="http://www.opengis.net/ows/2.0"
schemaLocation="http://schemas.opengis.net/ows/2.0/owsCommon.xsd"/>
  <xs:import namespace="http://www.opengis.net/3dps/1.0/core"
schemaLocation="http://www.opengis.net/3dps/1.0/core"/>
  <xs:complexType name="SceneLayerExtensionType">
    <xs:sequence>
      <xs:element name="AvailableFormat" type="ows:MimeType" minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element name="AvailableOffset" type="core:Position3DType" minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element name="AvailableOffsetMode" type="xs:string" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="GetSceneType">
    <xs:complexContent>
      <xs:extension base="core:AbstractGetPortrayalType">
        <xs:sequence>
          <xs:element name="Offset" type="core:Position3DType" minOccurs="0"
maxOccurs="1"/>
          <xs:element name="OffsetMode" type="ows:string" minOccurs="0" maxOccurs=
"1"/>
          <xs:element name="Format" type="ows:MimeType" minOccurs="1" maxOccurs="1"/>
          <xs:element name="Viewpoints" type="xs:string" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="ScenePortrayalCapabilitiesExtensionType">
    <xs:sequence>
      <xs:element name="SupportsArbitraryOffset" type="xs:boolean" minOccurs="0"
maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="SceneLayerExtension" type="scene:SceneLayerExtensionType"/>
</xs:schema>
```

13.3. view schema

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
```

```
targetNamespace="http://www.opengis.net/3dps/1.0/view">
  <xs:import namespace="http://www.opengis.net/ows/2.0"
schemaLocation="http://schemas.opengis.net/ows/2.0/owsCommon.xsd"/>
  <xs:import namespace="http://www.opengis.net/ows/2.0"
schemaLocation="http://schemas.opengis.net/ows/2.0/ows19115subset.xsd"/>
  <xs:import namespace="http://www.opengis.net/ows/2.0"
schemaLocation="http://schemas.opengis.net/ows/2.0/owsDomainType.xsd"/>
  <xs:import namespace="http://www.opengis.net/ows/2.0"
schemaLocation="http://schemas.opengis.net/ows/2.0/owsDataIdentification.xsd"/>
  <xs:import namespace="http://www.opengis.net/3dps/1.0/core"
schemaLocation="http://www.opengis.net/3dps/1.0/core"/>
  <xs:complexType name="ViewLayerExtensionType">
    <xs:sequence/>
  </xs:complexType>
  <xs:element name="Projection" type="view:ProjectionBaseType"/>
  <xs:complexType name="GetViewType">
    <xs:complexContent>
      <xs:extension base="core:AbstractGetPortrayalType">
        <xs:sequence>
          <xs:element name="BackgroundColor" type="xs:string" minOccurs="0"
maxOccurs="1"/>
          <xs:element name="TransparentBackground" type="xs:boolean" minOccurs="0"
maxOccurs="1"/>
          <xs:element name="Portrayals" type="PortrayalListType" minOccurs="1"
maxOccurs="1"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="PortrayalListType">
    <xs:sequence>
      <xs:element name="Portrayal" type="PortrayalType" minOccurs="1"
maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="PerspectiveProjectionType">
    <xs:complexContent>
      <xs:extension base="ProjectionBaseType">
        <xs:sequence>
          <xs:element name="POC" type="core:Position3DType" minOccurs="1"
maxOccurs="1"/>
          <xs:element name="POI" type="core:Position3DType" minOccurs="1"
maxOccurs="1"/>
          <xs:element name="UP" type="core:Position3DType" minOccurs="1" maxOccurs=
"1"/>
          <xs:element name="FOVX" type="xs:double" minOccurs="0" maxOccurs="1"/>
          <xs:element name="FOVY" type="xs:double" minOccurs="0" maxOccurs="1"/>
          <xs:element name="NearPlane" type="xs:double" minOccurs="0" maxOccurs="1"/>
          <xs:element name="FarPlane" type="xs:double" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
      </xs:extension>
```

```
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="ViewPortrayalCapabilitiesExtensionType">
    <xs:sequence>
      <xs:element name="farPlaneHint" type="xs:double" minOccurs="1" maxOccurs="1"/>
      <xs:element name="nearPlaneHint" type="xs:double" minOccurs="1" maxOccurs="1"/>
      <xs:element name="supportsMultipleViews" type="xs:boolean" minOccurs="1"
maxOccurs="1"/>
      <xs:element name="AvailableImageLayer" type="ImageLayerType" minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element name="AvailableProjection" type="AvailableProjectionType"
minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="OrthographicProjectionType">
    <xs:complexContent>
      <xs:extension base="ProjectionBaseType">
        <xs:sequence>
          <xs:element name="POC" type="core:Position3DType" minOccurs="1"
maxOccurs="1"/>
          <xs:element name="POI" type="core:Position3DType" minOccurs="1"
maxOccurs="1"/>
          <xs:element name="UP" type="core:Position3DType" minOccurs="1" maxOccurs=
"1"/>
          <xs:element name="Left" type="xs:double" minOccurs="0" maxOccurs="1"/>
          <xs:element name="Right" type="xs:double" minOccurs="0" maxOccurs="1"/>
          <xs:element name="Bottom" type="xs:double" minOccurs="0" maxOccurs="1"/>
          <xs:element name="Top" type="xs:double" minOccurs="0" maxOccurs="1"/>
          <xs:element name="NearPlane" type="xs:double" minOccurs="0" maxOccurs="1"/>
          <xs:element name="FarPlane" type="xs:double" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="ProjectionBaseType">
    <xs:sequence>
      <xs:element ref="ows:Identifier" minOccurs="1" maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="ImageLayerType">
    <xs:complexContent>
      <xs:extension base="ows:DescriptionType">
        <xs:sequence>
          <xs:element name="Identifier" type="ows:CodeType" minOccurs="1"
maxOccurs="1"/>
          <xs:element name="AvailableFormat" type="ows:MimeType" minOccurs="1"
maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
```

```
  <xs:complexType name="AvailableProjectionType">
    <xs:complexContent>
      <xs:extension base="ows:DescriptionType">
        <xs:sequence>
          <xs:element ref="ows:Identifier" minOccurs="1" maxOccurs="1"/>
          <xs:element name="ProjectionParameter" type="ProjectionParameterType"
minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="ProjectionParameterType">
    <xs:complexContent>
      <xs:extension base="ows:DomainType">
        <xs:sequence/>
        <xs:attribute name="required" use="optional" type="xs:boolean"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="PortrayalType">
    <xs:sequence>
      <xs:element name="Width" type="ows:int" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="Height" type="ows:int" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="ImageLayers" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="Formats" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="Qualities" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element ref="Projection" minOccurs="1" maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="PerspectiveProjection" type="view:PerspectiveProjectionType"
substitutionGroup="Projection"/>
  <xs:element name="OrthographicProjection" type="view:OrthographicProjectionType"
substitutionGroup="Projection"/>
</xs:schema>
```

13.4. info schema

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.opengis.net/3dps/1.0">
  <xs:import namespace="http://www.opengis.net/3dps/1.0/view"
schemaLocation="http://www.opengis.net/3dps/1.0/view"/>
  <xs:import namespace="http://www.opengis.net/ows/2.0"
schemaLocation="http://schemas.opengis.net/ows/2.0/owsCommon.xsd"/>
  <xs:import namespace="http://www.opengis.net/3dps/1.0/core"
schemaLocation="http://www.opengis.net/3dps/1.0/core"/>
  <xs:complexType name="InfoLayerExtensionType">
    <xs:sequence/>
    <xs:attribute name="queryable" use="optional" type="xs:boolean"/>
```

```
  </xs:complexType>
  <xs:complexType name="AbstractGetFeatureInfoType" abstract="true">
    <xs:complexContent>
      <xs:extension base="core:RequestBaseType">
        <xs:sequence>
          <xs:element name="Layers" type="xs:string" minOccurs="1" maxOccurs="1"/>
          <xs:element name="FeatureCount" type="xs:positiveInteger" minOccurs="0"
maxOccurs="1"/>
          <xs:element name="IdOnly" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
          <xs:element name="Format" type="ows:MimeType" minOccurs="1" maxOccurs="1"/>
          <xs:element name="Exceptions" type="ows:MimeType" minOccurs="0"
maxOccurs="1"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="GetFeatureInfoByRayType">
    <xs:complexContent>
      <xs:extension base="AbstractGetFeatureInfoType">
        <xs:sequence>
          <xs:element name="CRS" type="xs:anyURI" minOccurs="1" maxOccurs="1"/>
          <xs:element name="Height" type="xs:positiveInteger" minOccurs="1"
maxOccurs="1"/>
          <xs:element name="Width" type="xs:positiveInteger" minOccurs="1"
maxOccurs="1"/>
          <xs:element name="Projection" type="view:ProjectionBaseType" minOccurs="1"
maxOccurs="1"/>
          <xs:element name="ImagePosition" type="core:Position2DType" minOccurs="1"
maxOccurs="1"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="GetFeatureInfoByPositionType">
    <xs:complexContent>
      <xs:extension base="AbstractGetFeatureInfoType">
        <xs:sequence>
          <xs:element name="CRS" type="xs:anyURI" minOccurs="1" maxOccurs="1"/>
          <xs:element name="Coordinate" type="core:Position3DType" minOccurs="1"
maxOccurs="1"/>
          <xs:element name="Tolerance" type="xs:float" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="GetFeatureInfoByObjectIdType">
    <xs:complexContent>
      <xs:extension base="AbstractGetFeatureInfoType">
        <xs:sequence>
          <xs:element name="ObjectId" type="xs:anyURI" minOccurs="1"
maxOccurs="unbounded"/>
```

```
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:schema>
```

Chapter 14. Annex C: Tiling in scene-based 3D portrayal (informative)

Implementing tiling and refinement in scene-based portrayal

This standard intentionally does not define a way to delay-load tiles or other parts of a scene as the client is advancing within the portrayal. Part of the reason is that it is simply too early to describe the underlying techniques in sufficient detail to enable interoperability.

An often-used technique is to tile bigger scenes to achieve manageable chunks, and delay-load them as appropriate. This is often combined with refinement of geometries and/or image material, and performed using preprocessing.

This specification avoids assumptions about how a valid response looks like where possible. As a consequence, is is possible to answer to a request with a collection of LOD nodes acting as a spatial index that control the delay-loading of scene parts.

Such scenes may then refer to service invocations using core:GetResourceById operation invocations or resources defined outside of the service delivering the scene. That is, a delay-loading regime has to be mapped to the provisions of the format and then executed by causing delay-loading in the client. Intentionally, there is no limitation regarding the underlying methods.

However, there is the deliveryOption mechanism (DeliveryOption) which shall be used to avoid exposing uncommon practises to clients which are not prepared to handle them.

Chapter 15. Annex D: X3D (informative)

Extensible 3D (X3D)

The data comprising a 3D scene portrayal may come from different data sources and layers. For example, modern 3D geo-visualizations commonly include both the landscape of the real world and real-world objects such as trees, buildings. For real-time portrayal, these resources must be organized and composed into a data structure that can be rendered at at least 24 frames per second and respond to user input. The most well-known and widely adopted of the Web 3D technologies are the ISO set of languages, specifically the international standards of Virtual Reality Modeling Language (VRML) and its successor, Extensible 3D (X3D).

Extensible 3D (X3D) is an open and royalty-free International Standard for the description of such portrayals (ISO/IEC IS 19775-1:2013 http://www.web3d.org/documents/specifications/19775-1/V3.3/index.html). X3D generally refers to a suite of standards developed by the not-for profit Web3D Consortium (web3d.org). An X3D scene graph can be equivalently encoded as XML (ISO/IEC 19776-1), utf-8 (ISO/IEC 19776-2) and binary (ISO/IEC 19776-3). Runtime X3D scene graphs can be manipulated programmatically through the Scene Access Interface (SAI; ISO/IEC 19775-2). This API is bound to several languages including the standards for ECMAScript (ISO/IEC 19777-1) and Java (ISO/IEC 19777-2).

These scene graph languages are quite expressive in that a small number of basic elements (nodes) can be combined according to rules (content model) to create innumerable permutations and visualization possibilities. Consider the fundamental case of terrain: TIN-type geometry's explicit triangulation means more data to transmit, but it is faster to load and render in the client. The GRID-type structure is more compact to transmit because its connectivity is implicit, but it must be calculated to triangles on the client before rendering. These two types of geometry data structures are both supported by X3D portrayal, each in several forms. The TIN geometry type can be directly represented as an IndexedFaceSet (or IndexedTriangleSet), while the GRID-type can be represented by the ElevationGrid or the GeoElevationGrid. Authors must use the application and interface requirements to determine the composition of their scene for real-time delivery and portrayal.

One important set of nodes is defined in the Geospatial Component (Clause 25, http://www.web3d.org/documents/specifications/19775-1/V3.3/Part01/components/geodata.html).
The Geospatial component includes conventions that are defined by the Spatial Reference Model (see ISO/IEC 18026) including support for 23 standard ellipsoids and a number of nodes that can use spatial reference frames for modeling purposes. These new standard components enable applications to handle double precision coordinates and geometry in different projections as well as to manage multiple levels of detail (LODs) of geospatial data. The spatial reference frames supported by X3D 3.3 are geocentric, geodetic and UTM. The following nodes comprise the Geospatial component:

- GeoCoordinate
- GeoElevationGrid
- GeoLocation
- GeoLOD
- GeoMetadata

- GeoOrigin
- GeoPositionInterpolator
- GeoProximitySensor
- GeoTouchSensor
- GeoTransform
- GeoViewpoint

The Geospatial component allows for the mashup of responses from different services in that common coordinates can be used the GetScene response. The X3D specification requires that the client software perform the conversion to Cartesian computer graphics coordinates, subtracting any offset required to gain precision as specified in the GeoOrigin node. The Geospatial nodes also help create better interactive experiences in the 3D viewing client.

One important concept to note is that X3D is a modular standard when it comes to conformance. Related nodes are grouped into Components, which are detailed in each Clause of the specification. Components can be combined and supported at different levels of conformance, standardized as Profiles. Profiles enable tool builders to only implement the subset of the language they need for their application. In addition, X3D scenes contain header statements that designate the required node set to load the content, for example:

source~ PROFILE Interactive COMPONENT Geospatial source~

Table 59. summarized from the Annex of X3D 3.3 19775-1:

Core profile	**Absolute minimal file definitions required by X3D**
Interchange profile	Exchange of geometry and animations between authoring systems
Interactive profile	Implementing a lightweight playback engine that supports rich graphics and interactivity MPEG-4 interactive profile Providing the base point of interoperability with the MPEG-4 standard
CADInterchange profile	Distillation of computer-aided design (CAD) data to downstream applications, appropriately supporting Geometry and Appearance capabilities data for CAD
MedicalInterchange profile	Exchange of polygonal geometry, volumetric data and accompanying documentation between medical imaging systems
Immersive profile	Implementing immersive virtual worlds with complete navigational and environmental sensor controls
Full profile	The Full profile of X3D is comprised of all features of the standard

15.1. REFERENCES

[1] Reddy, M., Iverson, L., Leclerc, Y.G.: Under the hood of GeoVRML 1.0. In: Proceedings of the fifth symposium on Virtual reality modeling language (Web3D-VRML), pp. 23-28 (2000)

[2] McCann, M., Puk, R., Hudson, A., Melton, R., Brutzman, D.: Proposed enhancements to the x3d

geospatial component. In: Proceedings of the 14th International Conference on 3D Web Technology, pp. 155-158 (2009)

[3] Yoo, B., Brutzman, D.: X3D earth terrain-tile production chain for georeferenced simulation. In: Proceedings of the 14th International Conference on 3D Web Technology, pp. 159-166 (2009)

[4] Oliveira, N., Rocha, J.G.: Web 3D Service Implementation. In: Computational Science and Its Applications-ICCSA, pp. 538-549 (2013)

[5] Oliveira, N., Rocha, J.G.: Tiling 3d terrain models. Computational Science and Its Applications-ICCSA, pp. 550-561 (2013)

[6] Reitz, T., Krämer, M., Thum, S.: A processing pipeline for X3D earth-based spatial data view services. In: Proceedings of the 14th Int. Conf. on 3D Web Technology, pp. 137-145 (2009)

Chapter 16. Annex E: Revision History

Table 60. Document revision history.

Date	Release	Editor	Paragraph modified	Description
09/2013	-	Simon Thum	all	Initial revision
10.02.2014	-	Benjamin Hagedorn	Scope, overview, abbreviations	continued editing
16.02.2014	-	Benjamin Hagedorn	all	Updated to latest document template
through 2014	-	all	all	Various (see git history)
12.12.2014	-	Simon Thum	all	Prepare for OAB review
02.01.2015	-	Mike McCann	Annex D	Added Annex on X3D Geospatial
05/2015	-	Simon Thum	all	Prepare for submission to SWG
07/2015	-	Simon Thum Benjamin Hagedorn	all	Prepare for OAB review
27.08.2015	-	Simon Thum	all	Version for public comments
18.02.2016	-	Benjamin Hagedorn	all	Prepare for submission to TC
08.03.2016	-	Benjamin Hagedorn	all	Minor changes (update UML diagrams)
11.08.2016	-	Benjamin Hagedorn Ralf Gutbell	all	Prepare for publishing
17.03.2017	-	Benjamin Hagedorn	all	Prepare for publishing
12.09.2017	-	Benjamin Hagedorn	all	Switch format to OGC AsciiDoctor template