Open Geospatial Consortium Inc.

Date: 2015-08-27

Reference number of this OGC™ project document: **OGC 15-001r3**

OGC Version: 1.0.0

Category: OGC™ Implementation Standard

Editors: Benjamin Hagedorn, Simon Thum

3D Portrayal Service Standard

Contents

i. Abstract

The 3D Portrayal 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 Section 6.3.

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 standardisation, their technical content should be merged as far as possible. This document represents the efforts by the 3D Portrayal 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):

Organization	**Point of contact**
Fraunhofer Gesellschaft	Simon Thum
Hasso Plattner Institute at the University of Potsdam	Benjamin Hagedorn
Esri R&D Center Zürich	Thorsten Reitz

v. Submitters

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

Table 1: Editors

Name	**Affiliation**
Simon Thum	Fraunhofer IGD
Benjamin Hagedorn	Hasso Plattner Institute at the University of Potsdam
Thorsten Reitz	Esri R&D Center Zürich

Table 2: Contributors

Name	**Affiliation**
Volker Coors	Fraunhofer IGD, HFT Stuttgart
Arne Schilling	virtualcitySYSTEMS
Thomas H. Kolbe	Berlin University of Technology
Dieter Hildebrandt	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
Jan Klimke	Hasso Plattner Institute at the University of Potsdam

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 adresses use cases such as 3D portrayal of geodata and requesting additional information at the user's discretion. More details about possible interoperability scenarions 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.

2 Conformance

This OGC interface standard targets at 3DPS 1.0 implementations, i.e. 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 *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.

3 Normative References

The following normative documents contain provisions that, through reference in this text, constitute provisions of this document. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. For undated references, the latest edition of the normative document referred to applies.

WSC
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)*

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

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

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.

For the purposes of this document, the following additional 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.

5 Conventions

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

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

The term "3D scene" refers to a digital representation of geographic data that mainly composed from 3D graphics 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

Most of the abbreviated terms listed in Sub-clause 5.3 of the OGC Web Services Common Standard [OGC 06-121r9] apply to this document, plus the following abbreviated terms.

3DPS	3D Portrayal Service
UML	Unified Modeling Language
W3DS	Web 3D Service
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 OGC Web Service Common [OGC 06-121r9].

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 [OGC 06-121r9]. The contents of these data dictionary tables are normative, including any table footnotes.

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

6 3D Portrayal Service Overview

6.1 Overview

The 3D portrayal service standard (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 detail information like simulation results or other 3D spatial information provided using the service instances.

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]. OGC-internally, the development of 3D portrayal capabilities led to the proposals Web 3D Service (W3DS, serving 3D graphics data) and Web Perspective View Service (WPVS, serving rendered image data), which was generalized and published as OGC Web View Service (WVS, 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 working group recognizes the potential of tiling and welcomes work on standardisation to support tiling orthogonally to this standard.

In particular, any kind of approach that is based an 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 though OGC Web Service Common, the portable negotiation of format-specific idiosyncrasies and the seamless transition to server-side rendering offered by 3DPS. Several such formats are standardised, for example [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 standarized as OGC Web Service Common (WSC) based requests in that they provide potential for interoperability, do not limit the possible implemetations 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 3D scenes as a flow of geometries could not be expected to be amenable to standardisation efforts at the time.

At the same time, refinement of scenes or 3D (building) representations beyond multiple representations-based approaches were not in the base discussion papers and had to be left out due to 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 standardisation 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 standardise 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
- favours 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 this set represents a good way to a useful first interface supporting interoperable service-based 3D portrayal. The goal of 3DPS is to find out if actual interoperation is possible, i.e. to automate matching 3D portrayal clients to services.

6.4 Interoperability scenarios

The interoperability scenarios this standard enables mirror those demonstrated in the 3DPIE (see OGC 12-075), namely:

- Linking W3DS and WVS (Experiment 2 [1]; Section 7.3.2.2)
- Integration of multiple W3DS in a 3D client (Experiment 3; Section 7.3.2.3)
- W3DS/WVS for browser-based portrayal (Experiment 4; Section 7.3.2.4)
- W3DS/WVS for mobile portrayal (Experiment 5; Section 7.3.2.5)

It is expected that more interoperability scenarios are possible, but these are the practically proven scenarios.

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 performed by a 3DPS service.

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

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

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. *GetFeatureInfo* — This operation allows a client to retrieve more information about portrayed features.

[1] Experiment 1 is out of scope

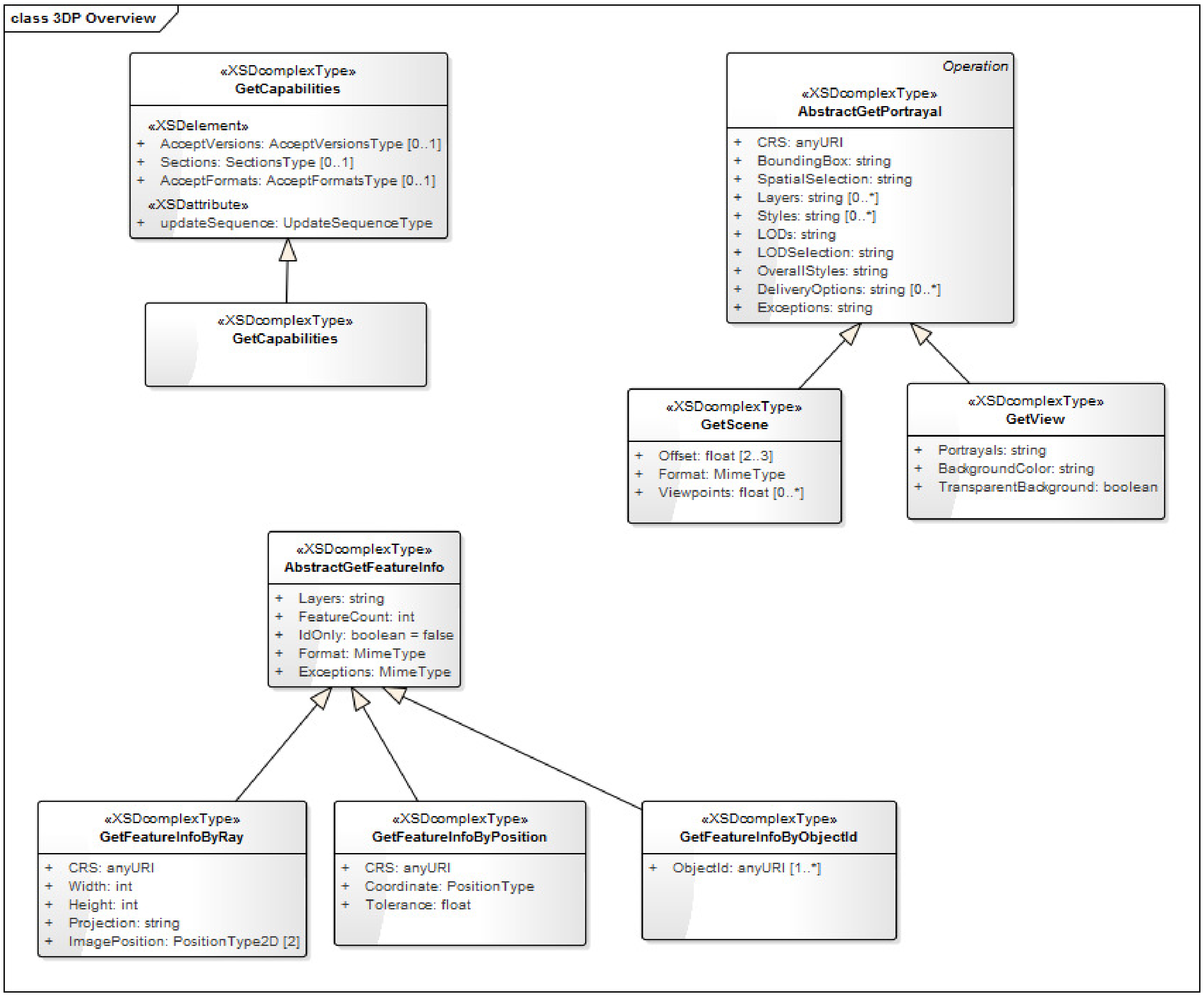

Figure 1: 3DPS UML class diagram

Figure 1 is a simple UML diagram summarizing the 3DPS interface. This class diagram shows that the 3DPS interface class inherits the GetCapabilities operation from the OGCWebService interface class 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 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, he 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 *AbstractGetPortrayal*, the *GetScene* and the *GetView* request types make use of the `RequestBase` structure which mimics the OWS Common [06-121r9] `RequestBase` data structure with the following adoptions, as shown in Figure 2 and Table 4:

- Attribute `service` contains the 3DPS service name, which is fixed to the string “3DPS”.
- Attribute `version` contains the 3DPS version number, which is fixed to the string “1.0”.

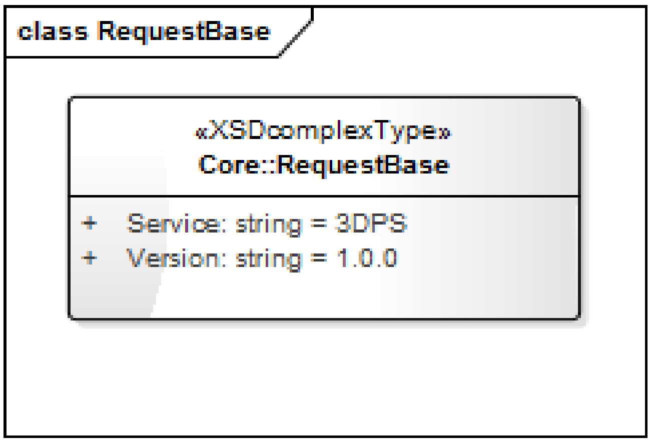

Figure 2: 3DPS `RequestBase` data structure UML class diagram

Table 4: 3DPS `RequestBase` components.

Name	**Definition**	**Data type**	**Multiplicity**
service Service	Service name	`String`, fixed to "3DPS"	one (mandatory)
version Version	Standard version for operation	`String`, fixed to "1.0"	one or more (mandatory)

All 3DPS requests, except *GetCapabilities*, **shall** use a data structure which is a subtype of `RequestBase`.

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".

7.3 Coordinate Reference Systems (CRS)

This standard uses several CRS types. 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 bouding 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 CRSes.
Viewpoint CRS	The viewpoint CRS is the CRS in which a viewpoint is defined.

7.3.1 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 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 behaviour potentially ensuing.

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

8 3DPS Core

8.1 Shared aspects

8.1.1 Position2D data structure

As defined in Table 6 , Position2D consist of two 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 6: Parameters in Position2D data structure

Names	**Definition**	**Data type and values**	**Multiplicity and use**
x1 X1	First position coordinate, in request CRS	Number type Origin and units specified by CRS	One (mandatory)
x2 X2	Second position coordinate, in request CRS	Number type Origin and units specified by CRS	One (mandatory)

8.1.2 Position3D data structure

As defined in Table 7 , Position3D consist 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: Parameters in Position3D data structure

Names	**Definition**	**Data type and values**	**Multiplicity and use**
x1 X1	First position coordinate, in request CRS	Number type Origin and units specified by CRS	One (mandatory)
x2 X2	Second position coordinate, in request CRS	Number type Origin and units specified by CRS	One (mandatory)
x3 X3	Third position coordinate, in request CRS	Number type Origin and units specified by CRS	One (mandatory)

8.2 GetCapabilities operation (mandatory)

A *GetCapabilities* operation, as required by OWS Common [OGC 06-121r9], allows a 3DPS client to retrieve service and scene metadata offered by a 3DPS server.

A core:GetCapabilities request **shall** consist of a core:GetCapabilities structure as defined in Figure 3 and Table 8.

8.2.1 GetCapabilities Request

The mandatory GetCapabilities operation allows clients to retrieve service metadata from a service. The response to a GetCapabilities request shall be an XML document containing service metadata about the service, including specific information about layers and portrayal capabilities. This clause specifies the XML document that a 3DPS instance shall return to describe its capabilities and contents.

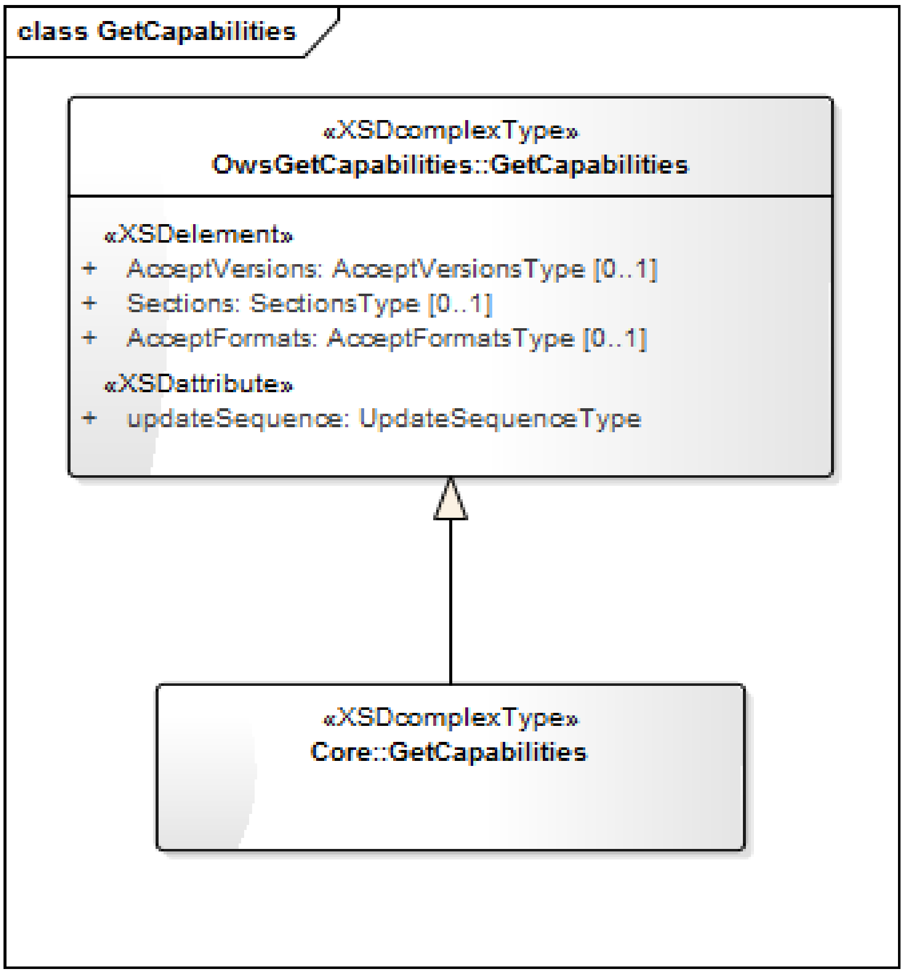
class GetCapabilities
«XSDcomplexType»
OwsGetCapabilities::GetCapabilities
«XSDelement»
+ AcceptVersions: AcceptVersionsType [0..1]
+ Sections: SectionsType [0..1]
+ AcceptFormats: AcceptFormatsType [0..1]
«XSDattribute»
+ updateSequence: UpdateSequenceType
«XSDcomplexType»
Core::GetCapabilities

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

Table 8: 3DPS core:GetCapabilities request components

Name	Definition	Data type	Multiplicity
service Service	Service name	Character String type, fixed to "3DPS"	one (mandatory)

8.2.2 GetCapabilities Response

A 3DPS metadata document consists of metadata as defined in Section 7.4.2 of OWS Common [OGC 06-121r9] section 7.4.2, a Contents section, and a PortrayalCapabilities section.

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

The response to a successful *core:GetCapabilities* request **shall** be based on a core:Capabilities structure as defined in Figure 4, Table 9, Figure 5, Table 10 and Table 13.

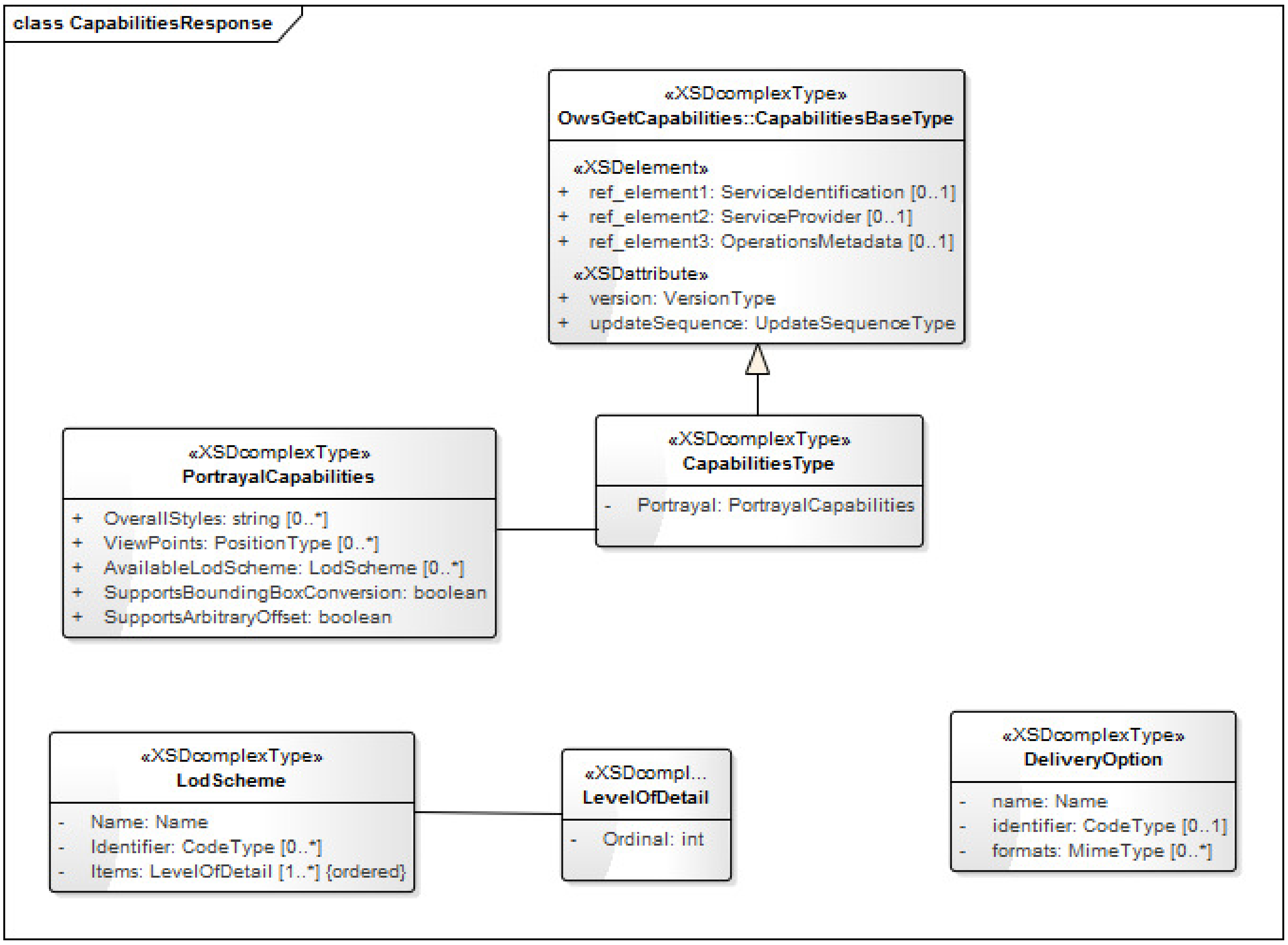
class CapabilitiesResponse
«XSDcomplexType»
OwsGetCapabilities::CapabilitiesBaseType
«XSDelement»
+ ref_element1: ServiceIdentification [0..1]
+ ref_element2: ServiceProvider [0..1]
+ ref_element3: OperationsMetadata [0..1]
«XSDattribute»
+ version: VersionType
+ updateSequence: UpdateSequenceType
«XSDcomplexType»
PortrayalCapabilities
+ OverallStyles: string [0..*]
+ ViewPoints: PositionType [0..*]
+ AvailableLodScheme: LodScheme [0..*]
+ SupportsBoundingBoxConversion: boolean
+ SupportsArbitraryOffset: boolean
«XSDcomplexType»
CapabilitiesType
- Portrayal: PortrayalCapabilities
«XSDcomplexType»
LodScheme
- Name: Name
- Identifier: CodeType [0..*]
- Items: LevelOfDetail [1..*] {ordered}
«XSDcompl...
LevelOfDetail
- Ordinal: int
«XSDcomplexType»
DeliveryOption
- name: Name
- identifier: CodeType [0..1]
- formats: MimeType [0..*]

Figure 4: 3DPS Capabilities UML class diagram

Table 9: Components of the core:Capabilities structure

Name	Definition	Data type	Multiplicity
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 [OGC 06-121r9].	as defined in [OGC 06-121r9]	as defined in [OGC 06-121r9]

Table 9: (continued)

Name	Definition	Data type	Multiplicity
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 [OGC 06-121r9].	as defined in [OGC 06-121r9]	as defined in [OGC 06-121r9]
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 [OGC 06-121r9].	as defined in [OGC 06-121r9]	as defined in [OGC 06-121r9]
Contents	Information about the 3D data content offered through this service	Contents type	zero or one
PortrayalCapabilities	Information about the portrayal capabilities of this service	PortrayalCapabilities Type	zero or one

8.2.2.1 Contents

The Contents section provides details about data layers that can be requested for portrayal. Its structure is extended from the Contents definition in OWS Common [OGC 06-121r9] by extending types referenced in it.

If a service does not offer any discriminate layers, it should make a best effort to emit a default layer that summarizes its offerings.

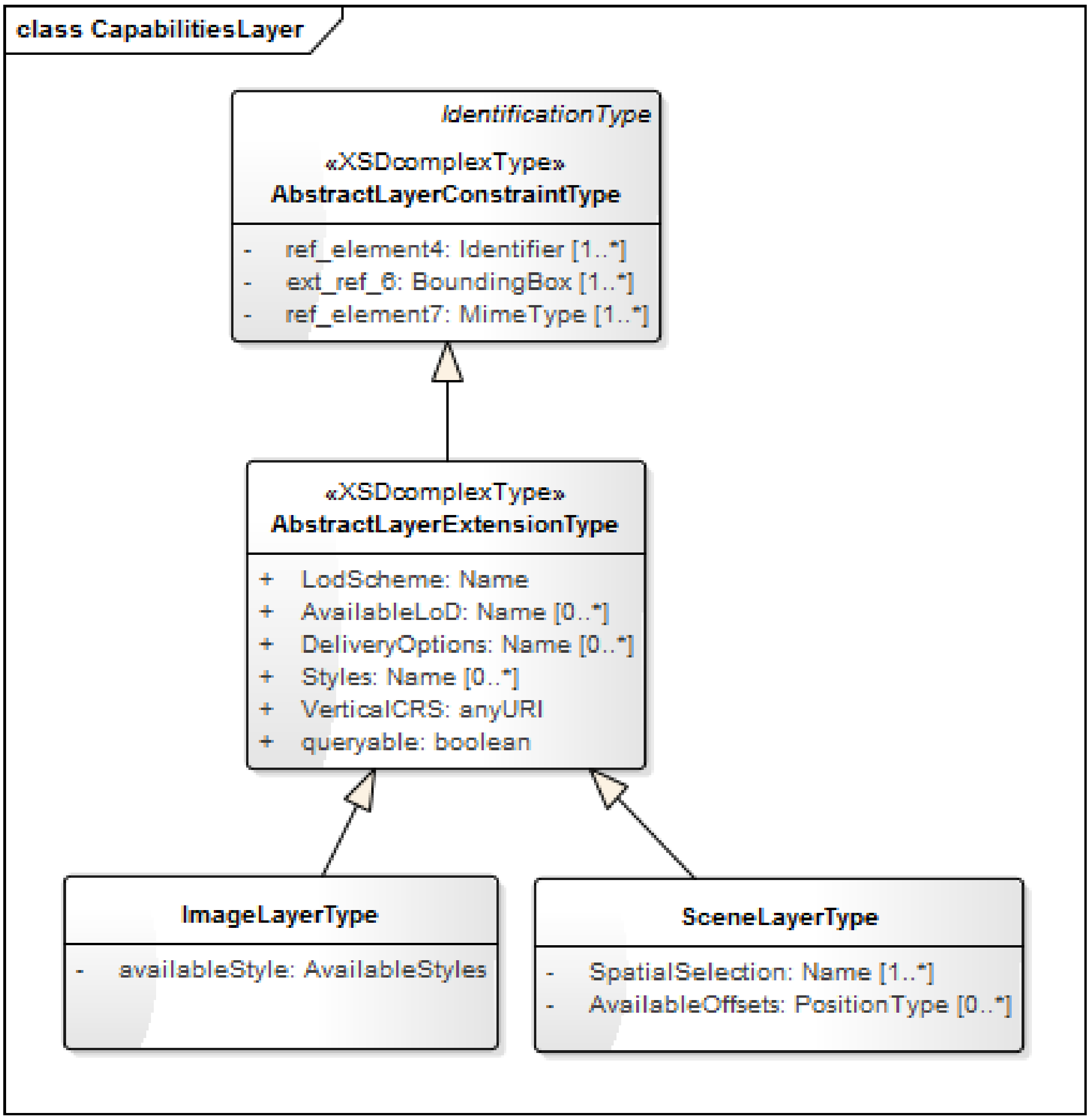
class CapabilitiesLayer
IdentificationType
«XSDcomplexType»
AbstractLayerConstraintType
- ref_element4: Identifier [1..*]
- ext_ref_6: BoundingBox [1..*]
- ref_element7: MimeType [1..*]
«XSDcomplexType»
AbstractLayerExtensionType
+ LodScheme: Name
+ AvailableLoD: Name [0..*]
+ DeliveryOptions: Name [0..*]
+ Styles: Name [0..*]
+ VerticalCRS: anyURI
+ queryable: boolean
ImageLayerType
- availableStyle: AvailableStyles
SceneLayerType
- SpatialSelection: Name [1..*]
- AvailableOffsets: PositionType [0..*]

Figure 5: 3DPS core:Contents and core:Layer and UML class diagram

Table 10: Components of the core:Contents structure

Name	Definition	Data type	Multiplicity
Layer	Metadata describing a data set available from this service	Layer data structure, see Table 11	Zero or more (optional) Include as much as layers shall be advertised

Table 11: Parts of core:Layer structure

Name	Definition	Data type	Multiplicity
title Title	Title of this dataset, human readable	`LanguageString`, see [OGC 06-121r9] clause 10.7	One (mandatory)
abstract Abstract	Brief narrative description of this dataset	`LanguageString`, see [OGC 06-121r9] clause 10.7	Zero or one (optional)
keywords Keywords	Unordered list of one or more commonly used or formalised word(s) or phrase(s) used to describe this dataset	`ows:Keywords` type	Zero or one (optional) One for each keyword authority used
identifier Identifier	Unambiguous identifier of this dataset, unique for this service	Character `String` type, not empty	Zero or one (optional) Include when may need to reference this dataset
metadata Metadata	Reference to more metadata about this layer	ows:Metadata, see [OGC 06-121r9], Table 35	Zero or more (include when useful)
wgs84BoundingBox WGS84BoundingBox	Minimum bounding rectangle surrounding dataset, specified in WGS 84 CRS with decimal degrees and longitude before latitude [a,c]	`OWS Common::WGS84BoundingBox`	Zero or more (optional) Include when useful or needed
boundingBox BoundingBox	Minimum bounding rectangle surrounding dataset, in available CRS [b,c]	`OWS Common::BoundingBox`	Zero or more (optional) Include when relevant and available, ideally at least one per available CRS
layer Layer	Metadata describing one subsidiary dataset available from this service	Layer data structure, see this Table	Zero or more (optional) One for each subsidiary Layer
availableCRS AvailableCRS	Coordinate reference system in which data from this layer may be requested	`URI`	One or more (mandatory)
availableLOD AvailableLOD	LOD values that hold data for this layer.	xs:Name	Zero or more (optional)
deliveryOption DeliveryOption	The delivery options available when requesting the layer. Empty means none are available.	xs:Name	Zero or more (optional)
availableStyle AvailableStyle	Style available for this layer	Style data structure, see Table Table 12	Zero or more (optional)
extensions Extensions	Hook for layer extensions	Extensions type	zero or one (optional)

[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.

Table 12: Parts of core:Style structure

Name	Definition	Data type	Multiplicity
title Title	Title of this style, human readable	`LanguageString`, see [OGC 06-121r9] clause 10.7	One (mandatory)
abstract Abstract	Brief narrative description of this style	`LanguageString`, see [OGC 06-121r9] clause 10.7	Zero or one (optional)
keywords Keywords	Unordered list of one or more commonly used or formalised word(s) or phrase(s) used to describe this style	`ows:Keywords` type	Zero or one (optional) One for each keyword authority used
identifier Identifier	Unambiguous identifier of this style, unique for this service	Character `String` 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 “true”

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 <CRS> element that is either stated explicitly or inherited from a parent Layer. The root <Layer> element shall include a sequence of zero or more CRS 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 <CRS> elements, each of which contains only a single CRS name.

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

AvailableLOD

Each layer may advertise a set of “levels of detail” present on that layer. Usually they are organized in one or more LOD Schemes, see AvailableLODScheme. The LOD scheme describes attributes to the individual levels of detail so a client is able to process and use them adequately.

Each LOD is described in a separate LOD node containing a unique identifier, title, description, and the actual value or magnitude of the LOD. This value is a URI consisting of a prefix and the actual numeric value, separated by a colon, e.g. "CityGML:1". The prefix indicates the spectrum of possible values and how these values should be interpreted, and conventionally is fixed per LODScheme node.

The numeric value indicates the actual “level” of detail on that ordinal scale. The scale values have a total order, which is connexive ($a > b$ 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 LODScheme node is not defined, but it is recommended to use an order increasing by the LODValue, from lower to higher levels of detail.

AvailableStyle

Each layer may advertise layer-specific styles (service styles) which modify the appearance of feature representations retrieved through the 3DPS AbstractGetPortrayal operation. The style’s Identifier is used in an AbstractGetPortrayal request parameter Styles. If only a single style is available, that style is known as the “default” style and does not need to be advertised by the service. The data structure of Style is listed in Table 12. 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.

Extensions

Component Extensions 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.

8.2.2.2 PortrayalCapabilities

The PortrayalCapabilities UML is included in Figure 4.

Table 13: 3DPS core:PortrayalCapabilities components

Name	Definition	Data type	Multiplicity
deliveryOptions DeliveryOptions	Delivery options supported when requesting this layer	Non-empty list of DeliveryOption type, see Table 14	One or more (optional)
availableLODScheme AvailableLODScheme	Name of the LOD scheme this layer supports	xs:string	One (mandatory)
supportsBoundingBoxConversion SupportsBoundingBoxConversion	Flag indicating if the Server can convert non-advertised BoundingBoxes	Boolean type (true/false)	Zero or one (optional)
backgrounds Backgrounds	Metadata describing backgrounds that can be used for portrayal generation	Non-empty list of Background type, see Table 15	One or more (optional)
viewpointHints ViewpointHints	Metadata describing meaningful viewpoints and settings of the virtual camera	Non-empty list of ViewPoint type, see Table 16	One or more (optional)
extensions Extensions	Hook for layer extensions	Extensions type	zero or one (optional)

Table 14: 3DPS core:DeliveryOptions

Name	Definition	Data type	Multiplicity
name Name	xs:Name	The name used to refer to the delivery option	One (Mandatory)
identifier Identifier	ows:CodeType	A code representing the delivery option.	One (mandatory)
format Format	ows:MimeType	A format which supports the delivery option. Empty means no restriction.	Zero or one (optional)

Table 15: Parts of core:Background structure

Name	Definition	Data type	Multiplicity
title Title	Title of the background, normally used for display to a human	Language String data structure, see [OGC 06-121r9] clause 10.7	One (mandatory)
abstract Abstract	Brief narrative description of this background, normally available for display to a human	Language String data structure, see [OGC 06-121r9] clause 10.7	Zero or one (optional)
identifier Identifier	Unambiguous identifier of this background, unique for this 3DPS service	Character String type, not empty	Zero or one (optional) Include when may need to reference this dataset

Table 16: Parts of core:Viewpoint structure

Name	Definition	Data type	Multiplicity
title Title	Title of the viewpoint, normally used for display to a human	Language String data structure, see [OGC 06-121r9] clause 10.7	One (mandatory)
abstract Abstract	Brief narrative description of this background, normally available for display to a human	Language String data structure, see [OGC 06-121r9] clause 10.7	Zero or one (optional)
identifier Identifier	Unambiguous identifier of this background, unique for this 3DPS service	Character String type, not empty	Zero or one (optional) Include when may need to reference this dataset
viewpointComponents	Camera and projection specification for generating the desired portrayal	ProjectionBaseType, not empty E.g., PerspectiveProjection	OrthographicProjection, see

DeliveryOptions

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. Similarly, whether it sends a color image or relative depth from a given viewpoint might be controlled this way.

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 a given 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 gives deliver options names and URIs that 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, if a client may communicate and portray properly data from a set or subset of service instances should be know in advance, and not through user-observed or silent failure. 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 mechanisam to do that, delivery options enable safeguarding the development of improved mechanisms so 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. Dropping back to naming profiles on MIME types which are to be standardised as well to provide scope for interoperability.

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.

AvailableLODScheme

The LODScheme child node describes a set of Levels of Detail that can be provided by the layer (see Figure 5). A layer may contain objects in several representations to choose from. In this document, the term LOD always 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 as simple box geometry, as a geometry with additional façade textures, as a group containing elements for walls, roofs, windows, doors, or even 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 his scene graph from subsets of the same layer having different LODs and thus adjust the workload placed on the graphics pipeline.

Each LOD scheme (of which there may be one or more) defines a full order of LOD definitions. 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.

Informative: The commonly understood LOD definitions of CityGML can be found in [08-007r1] clause 6.2, and are specified in Table 20.

SupportsBoundingBoxConversion

True if the service instance supports conversion of bounding boxes from the request CRS into appropriate layer CRSes for query.

Backgrounds

Backgrounds is a non-empty list containing Background elements as described in Table 15.

Viewpoints

The parameter value contains a comma separated list of tuple values for defining for each viewpoint a) a name, b) the Point of Interest (POI), c) the Point of Camera (POC), d) the up vector (UP), and e) the Field of View (FOV). Thus 11 values are used to define one viewpoint. In case of multiple viewpoints, all values are appended to the list consecutively. The list of Viewpoints parameter values shall be divisible by 11.

> Template:
> Viewpoints=NAME,POIx,POIy,POIz,POCx,POCy,POCz,UPx,UPy,UPz,FOV

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

Extensions

Component Extensions is provided as a canonical place for extensions to define service metadata and can be used, e.g., as a hook for operaton-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 Subclause 7.4.1 of [OGC 06-121r9].

8.3 Capabilities document XML encoding

A XML schema fragment for a service metadata document extends OWS CapabilitiesType in `owsCommon.xsd` of [OGC 06-121r9] as refined for the 3DPS, and may be reviewed under Section 13.

As indicated, this XML Schema Document uses the `owsServiceIdentification.xsd`, `owsServiceProvider.xsd`, and `owsOperationsMetadata.xsd` schemas specified in [OGC 06-121r9]. It also uses XML Schema Documents for the Contents and PortrayalCapabilities sections of the Capabilities XML document, which are split per target namespace as listed in Table 17. All these XML Schema Documents contain documentation of the meaning of each element, attribute, and type, and this documentation shall be considered normative as specified in Subclause 11.6.3 of [OGC 06-121r9].

Table 17: 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

8.3.1 Sample GetCapabilities request and response

EXAMPLE: A GetCapabilities request may look like this:

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

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

```
<?xml version="1.0" encoding="UTF-8"?>
<ps:Capabilities xmlns:ows="http://www.opengis.net/ows/2.0"  xmlns:ps="http://www.opengis. ↩
   net/3dps/1.0"
 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=" ↩
    1.0.0">
    <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>1.0.0</ows:ServiceTypeVersion>
        <ows:Profile>default</ows:Profile>
        <ows:Fees>none</ows:Fees>
        <ows:AccessConstraints>none</ows:AccessConstraints>
    </ows:ServiceIdentification>
    <ows:ServiceProvider>
        <ows:ProviderName>Fraunhofer IGD</ows:ProviderName>
        <ows:ProviderSite xlink:href="http://www.igd.fraunhofer.de/en/Institut/Abteilungen/ ↩
            GEO" />
        <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/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_blank</ows:Value>
                    <ows:Value>blank</ows:Value>
                    <ows:Value>errormarker</ows:Value> <!-- custom extension -->
                </ows:AllowedValues>
                <ows:DefaultValue>text/xml</ows:DefaultValue>
                <ows:Meaning>the type of exception report expected for a request</ ↩
                    ows:Meaning>
            </ows:Parameter>
            <ows:Parameter name="NoData">
                <ows:AllowedValues>
                    <!-- empty as an allowed value mirrors supportsNoData -->
                    <ows:Value>empty</ows:Value>
                    <ows:Value>blank</ows:Value>
                </ows:AllowedValues>
                <ows:DefaultValue>empty</ows:DefaultValue>
                <ows:Meaning>what to do if no data is present in the response</ows:Meaning>
            </ows:Parameter>
            <!-- the following are part of the standard and may be omitted -->
            <ows:Parameter name="crs">
                <ows:Meaning>the coordinate reference system the client expects the  ↩
                    response to be encoded in</ows:Meaning>
            </ows:Parameter>
        </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:Meaning>the type of exception report expected for a request</ ↩
                    ows:Meaning>
            </ows:Parameter>
            <ows:Metadata>
                <ows:AdditionalParameter>
                    <ows:Name>unspecifiedParameter</ows:Name>
                    <ows:Value></ows:Value>
                </ows:AdditionalParameter>
            </ows:Metadata>
        </ows:Operation>
        <ows:Parameter name="boundingBox">
            <ows:AnyValue/>
        </ows:Parameter>
        <!-- TODO list all parameters -->
    </ows:OperationsMetadata>
    <Contents>
        <Layers>
            <ows:Identifier>layer1</ows:Identifier>
            <ows:BoundingBox>
```

```
                <ows:LowerCorner></ows:LowerCorner>
                <ows:UpperCorner></ows:UpperCorner>
            </ows:BoundingBox>
            <ows:OutputFormat>text/xml</ows:OutputFormat>
            <ows:OutputFormat>model/x3d+xml</ows:OutputFormat>
            <LodScheme>CityGML</LodScheme>
        </Layers>
        <Layers>
            <ows:Identifier></ows:Identifier>
            <ows:BoundingBox>
                <ows:LowerCorner></ows:LowerCorner>
                <ows:UpperCorner></ows:UpperCorner>
            </ows:BoundingBox>
            <ows:OutputFormat>text/xml</ows:OutputFormat>
            <ows:OutputFormat>model/x3d+xml</ows:OutputFormat>
            <LodScheme>CityGML</LodScheme>
        </Layers>
    </Contents>
    <Portrayal>
        <OverallStyles>wireframe</OverallStyles>
        <OverallStyles>dynamicSky</OverallStyles>
        <OverallStyles>energyLossEstimation</OverallStyles>
        <LodScheme>
            <Name>CityGML</Name>
            <Identifier codeSpace="http://www.opengis.net/3dps/1.0">CityGML</Identifier>
            <Items>CityGML:0</Items>
            <Items>CityGML:1</Items>
            <Items>CityGML:2</Items>
            <Items>CityGML:3</Items>
            <Items>CityGML:4</Items>
        </LodScheme>
        <LodScheme>
            <Name>custom</Name>
            <Identifier codeSpace="http://example.com">ExampleCustomLod</Identifier>
            <Items>boundingBox</Items>
            <Items>simplified</Items>
            <Items>full</Items>
        </LodScheme>
        <SupportsNoData>true</SupportsNoData>
        <SupportsBoundingBoxConversion>true</SupportsBoundingBoxConversion>
    </Portrayal>
</ps:Capabilities>
```

8.4 AbstractGetPortrayal Operation (abstract)

The abstract `core:AbstractGetPortrayal` operation specifies the commonality between the two represented approaches, 3D scene-graph delivery and rendered image delivery. The actual operations (serving a particular approach) shall be defined by adding 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 6 and Table 18. This applies only to actual implementations of the request.

core:GetCapabilities structure

Figure 6: 3DPS `core:AbstractGetPortrayal` operation request UML class diagram

Table 18: Parameters of the AbstractGetPortayal operation.

Names	Definition	Data type and values	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 specified by each Implementation Standard and Schemas version	One (mandatory)
crs CRS	Primary CRS	`anyURI` as defined in [OGC 06-121r9] clause 10.3	One (mandatory)
boundingBox BoundingBox	Bounding box surrounding selected dataset, in available CRS.	`BoundingBox` data structure, see [OGC 06-121r9] clause 10.2.	*to be specified per operation*
spatialSelection SpatialSelection	Indicates method of selecting objects with BoundingBox	`String`, not empty. Value is one of “contains_center”, “overlaps”, “cut”. Default is “overlaps”.	*to be specified per operation*
layers Layers	List of layer to retrieve the data from	`StringList` type, comma separated list of layer Identifiers	One (mandatory)
styles Styles	List of service styles to be applied to the layers	`StringList`, comma-separated list of one rendering style per requested layer	Zero or one (optional)
background Background	Identifier of desired background	Character String type, not empty Values are specified in service metadata	Zero or one (optional) Include when background desired
lods LODs	List of LODs requested for the layers	`StringList`, comma separated list of `URIs`	Zero or one (optional)
lodSelection LODSelection	Indicates method for selecting LODs	`String`, Values are specified in service metadata	Zero or one (optional)
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	Names of the delivery options requested by the client	`StringList`, comma separated list of `URIs`	Zero or more (optional)
exceptions Exceptions	Format of exceptions	ows:MimeType, see [OGC 06-121r9] clause 10.5	Zero or one (optional)

8.4.1.1 CRS

The parameter value for the coordinate reference system (CRS) is defined in [OGC 06-121r9] clause 10.3 and [OGC 04-046r3]. If a 2D CRS is used, the height reference is taken from the layer’s vertical datum.

The given CRS is considered to be in effect for the operation query and response. In particular, it will be considered as the CRS for the bounding box if that is not specified explicitly. See Section 7.3 for details.

8.4.1.2 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 boundingBox CRSs other than available Layer CRS, this capability should be advertised in the Capabilities.

The `BoundingBox` data structure is defined in [OGC 06-121r9] clause 10.2. The units, ordering and direction of increment of the x1, x2 and x3 axes are as defined by the `CRS` parameter.

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 behaviour should be assumed.

8.4.1.3 SpatialSelection

The default selection method is that any features that are partly or entirely contained in the `BoundingBox` shall be returned.

If a spatial selection method other than intersecting the BoundingBox with the feature's spatial extent is desired, the SpatialSelection parameter can be used. The default behaviour of selecting a feature is to check the spatial relation between the BoundingBox and the feature's 3D geometry.

Table 19 defines the possible spatial selection modes.

Table 19: Spatial selection modes

Name	Description
overlaps 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 Cut	This spatial selection method shall not return features or part 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.

8.4.1.4 Layers

The Layers parameter specifies a comma separated list of layer identifiers 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 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 comma separated list shall refer to a layer identifier as described in the service's meta data.

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 serve all data available, only a subset of the data, or no data at all.

8.4.1.5 Styles

The mandatory `Styles` parameter lists the style in which each layer is to be rendered. The value of the `Styles` parameter is a comma-separated list of one or more valid style identifiers. There is a one-to-one correspondence between the values in the `Layers` parameter and the values in the `Styles` parameter. Each data layer in the list of `Layers` is rendered using the corresponding style in the same position in the list of STYLES. Each style Name shall be one that was defined for or inherited by this layer as specified in the service meta data. (In other words, the client may not request a layer in a style that was only defined for a different layer.)

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 the default style, a request shall contain either multiple comma-separated values one for each layer (as in “STYLES=„„”) or a single empty value (“STYLES=”).

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 capabilities.

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

Example:

Layers=dtm,buildings,vegetation&Styles=orthophoto„simple

8.4.1.6 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 comma separated list of names referring to the available Levels of Detail as specified in the service meta data’s layer section and portrayal capabilities section.

The length of this 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*.

Conventionally, a name consists of a prefix and the actual numeric LOD value, separated by a colon, e.g. “CityGML:4” for indoor models. The prefix indicates the spectrum of possible values and how these values should be interpreted.

Table 20 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.

Table 20: Well-known LOD names

URIs	**Description**
CityGML:0, CityGML:1, CityGML:2, CityGML:3, CityGML:4	OGC [08-007r1] clause 6.2
INSPIRE:0, INSPIRE:1, INSPIRE:2, INSPIRE:3, INSPIRE:4	same as citygml

8.4.1.7 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 layer. Four selection methods are predefined that fit the well-known LOD-based multiple representations approach; thus they are not necessarily sensible for a given implementation and should be checked to be supported in the `GetCapabilities` response before invocation.

The predefined LOD selection methods are defined in Table 21.

Table 21: 3DPS LOD selection modes

Name	Description
equals Equals	For each feature, the available LODs are compared with the LOD value provided in the AbstactGetPortrayal request. If the specified LOD is available for this feature, then the according model shall be selected and included for portrayal response. If the specified LOD is not available for this feature, then the feature shall not be included in the scene or view. 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 specified LOD is available for this feature, then the according model shall be selected and included in the scene. If the specified 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 specified 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 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 specified LOD value in the AbstactGetPortrayal request.
equal_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.

The selection methods offered by a 3DPS service shall be advertised in the service's meta data. They are not necessarily listed here as this is considered an extension point.

8.4.1.8 OverallStyles

The optional `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.

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 endornment 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. Therefore, if none are specified, an implementation should not deliver any. If the combination of styles requested cannot be delivered, the service should make an effort to prioritise the first-mentioned overall styles, or return an exception.

8.4.1.9 DeliveryOptions

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 DeliveryOptions 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.

8.4.1.10 Exceptions

The optional Exceptions parameter specifies the behaviour of the service upon detecting an error, e.g. invalid request or internal server error. The default value is “text/xml”. The value shall be one of the MIME types offered in the service’s Capabilities document meta data parameter value, see Section 8.3.1. This may include the special value “blank”, as outlined below.

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 document with a valid structure according to the format and no, or no useful, information 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 Section 8.4.3).

8.4.2 AbstractGetPortrayal Response

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

8.4.3 Exceptions

If a 3DPS service encounters an error while performing a GetCapabilities operation, it shall return an exception report message as specified in Clause 8 of [OGC 06-121r9].

In case of an incorrect request or an intermittent error while generating or delivering the response, the response shall be supplied in the requested format for exceptional cases (parameters 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 [06-121r9] remain valid. The exception codes listed in Table 22 are defined for more specific cases.

Table 22: 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 delivery option is requested	the delivery option name
`LodNotDefined`	an unadvertised LOD name is requested	the LOD name
`LodNotApplicable`	if 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 the OGC Web Services Common Standard [OGC 06-121r9] Section 9.3, subsequently the “base GetResourceByID operation”.[2]

[2] Following the notion that “Id” is an abbreviation not an acronym

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 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

This 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 23: 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 2: http://www.opengis.net/spec/3DPS/1.0/req/service/core/getResourceById/request

A `core:GetResourceById` request **shall** be conformant to the base GetResourceByID operation as defined in [OGC 06-121r9], Section 9.3, with the additional constraints laid out in this section.

Table 24: Changed parameters of the GetResourceById operation.

Names	**Definition**	**Data type and values**	**Multiplicity and use**
resourceId ResourceId [2]	Unambiguous identifier of desired resource	`URI`, not empty	One (mandatory)

8.5.3.1 ResourceId

The `resourceId` parameter shall be mandatory.

8.5.3.2 OutputFormat

The `outputFormat` parameter shall be mandatory and specify 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 3: 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.

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 3D scene based client wanting to request the geodata an 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 reply and established the instance's capability to serve compatible 3D geodata.

Existing and evolving 3D model formats and their delivery mechanisms vary a lot. Thus, 3DPS `GetScene` is intentionally defined to allow for a range of mechanisms, potentially limiting the scope for interoperability to quite specific client/server combinations. However, 3DPS `GetScene` intends 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 Capabilities document.

Requirement 4: 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 the `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 5: 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 service's OperationsMetadata 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 6: 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 Extension element as defined in Figure 7 and Table 25
Dependency: 3DPS Core (http://www.opengis.net/spec/3DPS/1.0/conf-class/core)

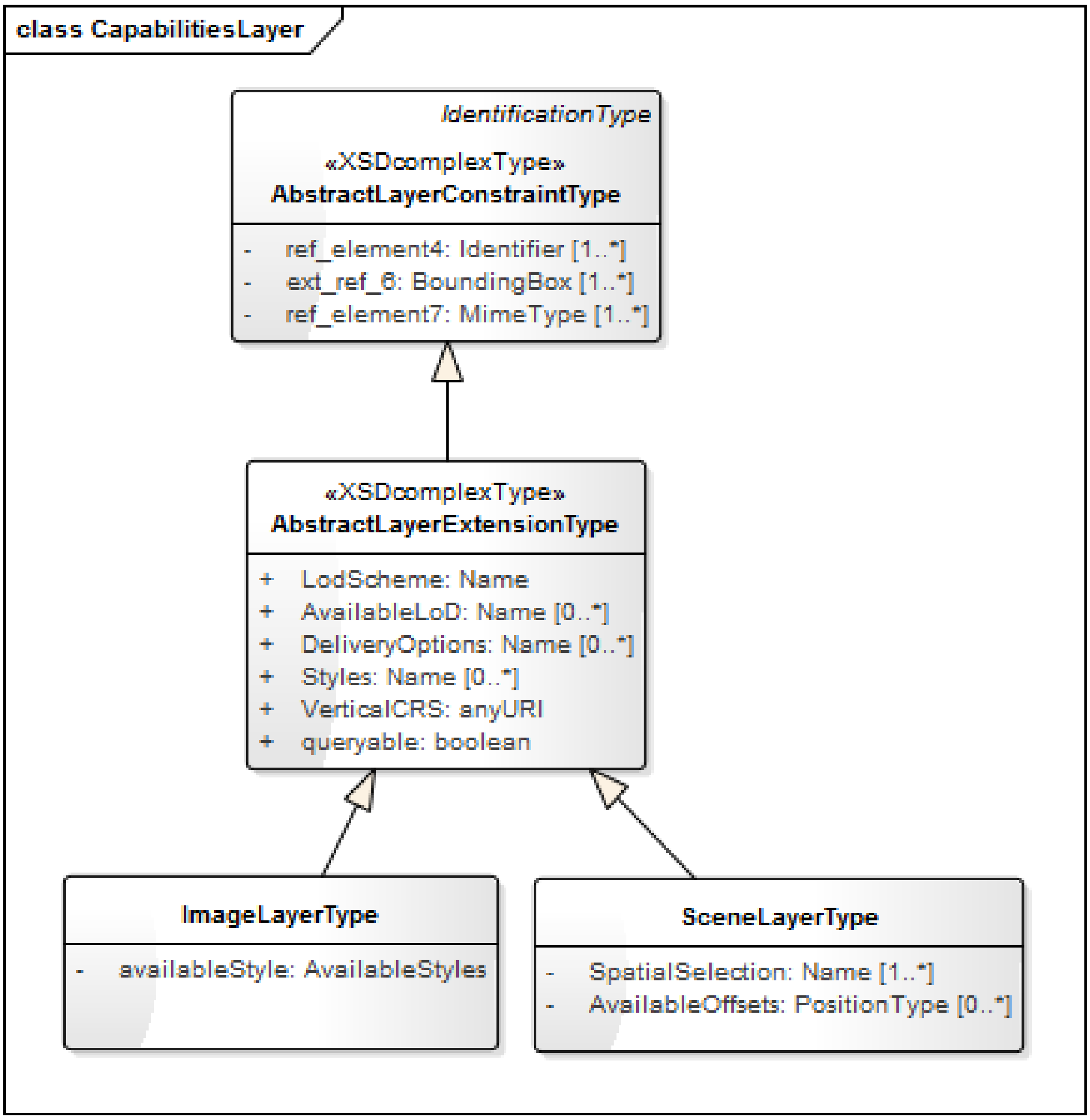
class CapabilitiesLayer
IdentificationType
«XSDcomplexType»
AbstractLayerConstraintType
- ref_element4: Identifier [1..*]
- ext_ref_6: BoundingBox [1..*]
- ref_element7: MimeType [1..*]
«XSDcomplexType»
AbstractLayerExtensionType
+ LodScheme: Name
+ AvailableLoD: Name [0..*]
+ DeliveryOptions: Name [0..*]
+ Styles: Name [0..*]
+ VerticalCRS: anyURI
+ queryable: boolean
ImageLayerType
- availableStyle: AvailableStyles
SceneLayerType
- SpatialSelection: Name [1..*]
- AvailableOffsets: PositionType [0..*]

Figure 7: 3DPS `scene:Layer` extensions UML class diagram

Table 25: Extension of the `core:Layer` structure.

Names	**Definition**	**Data type and values**	**Multiplicity and use**
availableStyle AvailableStyle	Style available for this layer	Style data structure, see Table TODO	Zero or more (optional)
availableFormat AvailableFormat	Output format valid for this layer	ows:OutputFormat type	Zero or more (optional)

Table 25: (continued)

Names	Definition	Data type and values	Multiplicity and use
availableOffset+ Available-Offset	Offsets available for querying this layer	FixedOffset data structure, see Table 27	Zero or more (optional)

9.2.3.1 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 proto types. The size of these symbols may be scaled according on feature attribute values. It is recommended to use the Symbology Encoding (SE) and Filter Encoding (FE) as basis for defining service styles (see [OGC 04-095] and [OGC 05-077r4]), and extend the capabilities for styling 3D objects (see [OGC 09-042]).

9.2.3.2 AvailableFormat

Information about data formats in which the advertised Layers are available is added to the 3DPS service metadata by using the layer extension mechanism provided by the 3DPS Core.

AvailableFormat items advertise available encodings of scene returned by the GetScene operation as ows:MimeType as per [OGC 06-121r9] clause 10.5.

Table 26 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 Geospatial component which is important to support as it allows for combination of content from different services.

Table 26: 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

Requirement 7: 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 in addition, shall advertise and recognize the MIME types named in Table 26 if the formats represented by them are supported, even if more qualified alternatives exist.

9.2.4 AvailableOffsets

Any fixed offsets in which the layer is available. A client should not expect other offsets to be available unless the arbitraryOffset 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, for example 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 geo coordinates, 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 27. For all offset modes it holds that CRS coordinate order is observed in the request. An implemetation 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 27 for increased interoperability.

Table 27: Offset modes

Name[a]	**Definition**
subtract Subtract	The offset is subtracted from all coordinates specifying absolute geolocations. Thus, when the client adds the offset to those again, true geocoordinates in the request CRS are obtained.
embed 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 wheter 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.
[a] The name capitalization rules being used here are specified in Subclause 11.6.2 of [OGC 06-121r9].	

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 Service Metadata by using the capabilities extension mechanism provided by the 3DPS Core.

9.2.6 Additions to PortrayalCapabilities structure

Requirement 8: http://www.opengis.net/spec/3DPS/1.0/req/service/scene/portrayalcapabilities-extension

A 3DPS service implementing conformance class *scene* shall extend the core:PortrayalCapabilities Extension element as defined in Figure 8, Table 28
Dependency: 3DPS Core (http://www.opengis.net/spec/3DPS/1.0/conf-class/core)

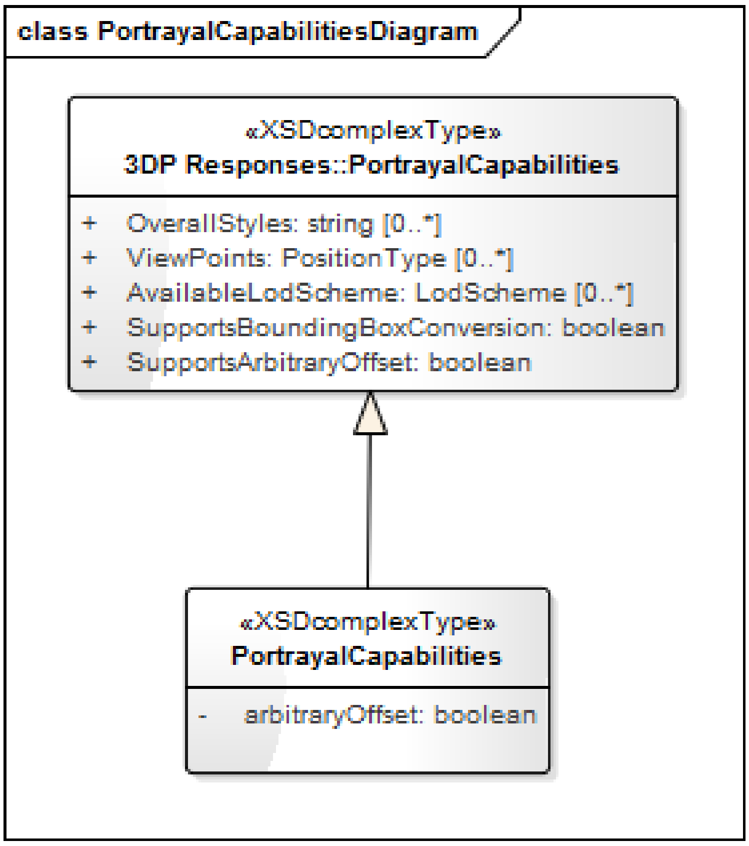
class PortrayalCapabilitiesDiagram
«XSDcomplexType»
3DP Responses::PortrayalCapabilities
+ OverallStyles: string [0..*]
+ ViewPoints: PositionType [0..*]
+ AvailableLodScheme: LodScheme [0..*]
+ SupportsBoundingBoxConversion: boolean
+ SupportsArbitraryOffset: boolean
«XSDcomplexType»
PortrayalCapabilities
- arbitraryOffset: boolean

Figure 8: 3DPS `scene:PortrayalCapabilities` UML diagram

Table 28: Extension of the `core:PortrayalCapabilities` structure.

Names	Definition	Data type and values	Multiplicity and use
arbitraryOffset ArbitraryOffset	Specifies whether the client may request arbitrary offset parameters	Boolean type, not empty Value either "true" or "false". Default is "false"	One (mandatory)

9.3 GetScene request

Requirement 9: 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 9 and Table 29.

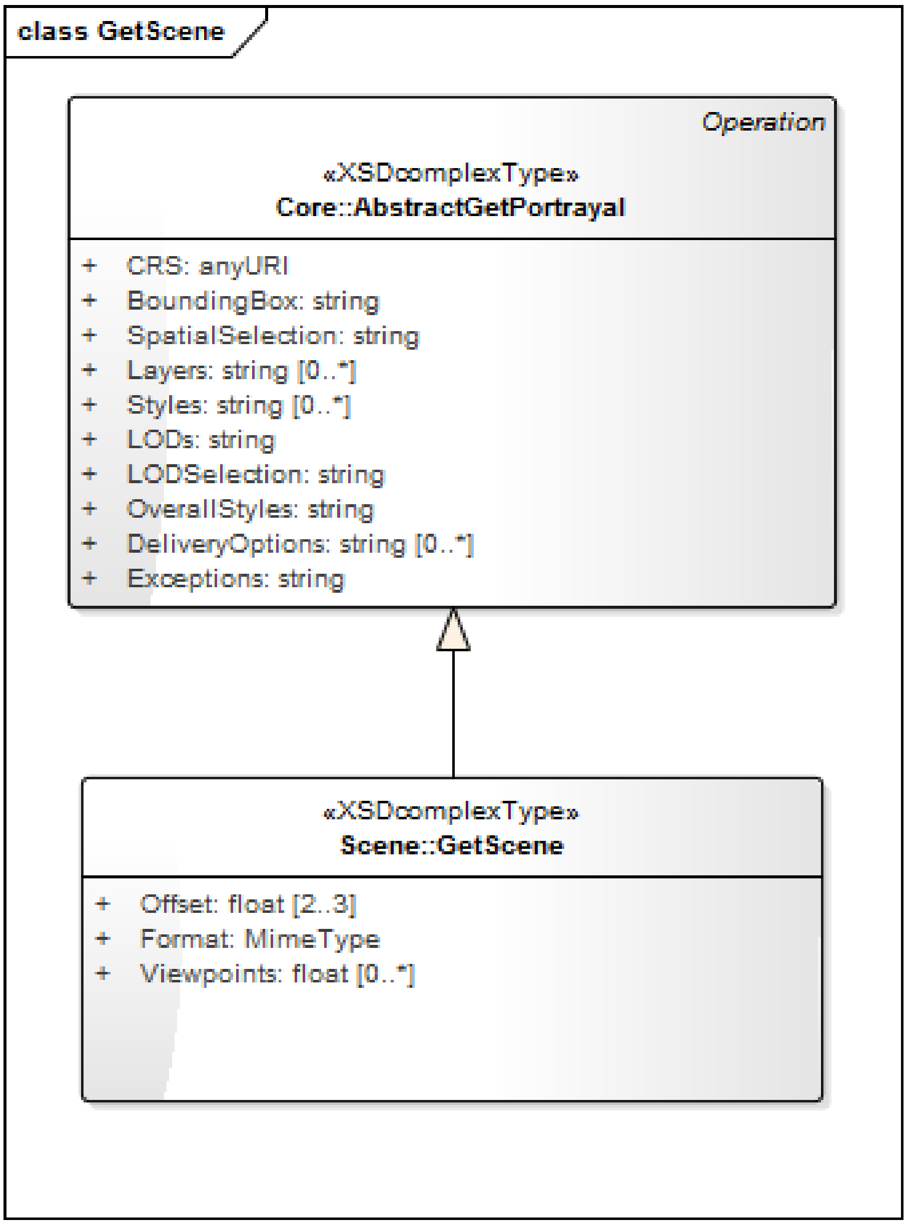

Figure 9: 3DPS `scene:GetScene` request UML class diagram

Table 29: 3DPS `scene:GetScene` request components

Names[a]	**Definition**	**Data type and values**	**Multiplicity and use**
offset Offset	Offset vector which shall be applied to the scene, using the mode specified (if any)	Character String type, list of coordinate value of length 2 or 3, separated by comma [b]. Optionally followed by a colon and an explicit offset mode name	Zero or one (optional)
format Format	Format encoding of the scene	ows:MimeType, see [OGC 06-121r9] clause 10.5	One (mandatory)
viewpoints Viewpoints	Add Viewpoints to choose from	Character String type, list of tuple value, separated by comma.	Zero or one (optional)
[a] a The name capitalization rules being used here are specified in Subclause 11.6.2 of [OGC 06-121r9]. [b] Coordinates are ordered in the request CRS order.			

9.3.1 Offset

The offset and, optionally, offset mode to use for scene delivery. See Section 9.2.4 and Section 9.2.5.

9.3.2 Format

The mandatory Format parameter specifies the target encoding of the returned scene provided as ows:MimeType, see [OGC 06-121r9] clause 10.5. Available formats are described in the service's meta data.

9.3.3 Viewpoints

The optional parameter Viewpoints can be used to include a list of viewpoints in the scene the 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.

9.3.4 Delivery Options

Typical delivery options in `GetScene` based requests may be obtained from Table 30.

Table 30: Additional scene:DeliveryOption names

Names	**URI**	**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 format, but generally not any actual geometries
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. The document contains a 3D scene assembled from the features of the selected `Layers` (mainly) within the specified `BoundingBox`, represented in the specified `CRS` and `Format`.

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 an overallStyle, 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 implemetation 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 implemetation chooses to treat it as an error, the error code should be `FormatCapabilityMissing`. The same holds when the service implemetation 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 maximise interoperability, URIs representing serivce 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 behaviour should be guarded by appropriate delivery options to make sure a general client does not request it by accident.

Requirement 10: 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 GetView request may return any of the exception reports defined in `core:AbstractGetPortrayal` and/or as specified in Clause 8 of [OGC 06-121r9].

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

The `GetScene` request my be issued as a HTTP Get KVP-based request.

9.6.1 HTTP/GET + KVP Binding

The folowing table defines the KVP encoding for `GetScene`. It contains the complete parameter mapping including the parameters defined by `AbstractGetPortrayal` and `core:RequestBase`.

Table 31: Parameters of the GetScene operation.

Names	Definition	Data type and values	Multiplicity and use
service Service	Service type identifier	`String`, fixed to “3DPS”	One (mandatory)
request Request	Operation name	`String`, Fixed to "GetScene".	One (mandatory)
version Version	Standard version for operation	`String`, not empty Fixed to "1.0".	One (mandatory)
crs CRS	Primary CRS	`anyURI` as defined in [OGC 06-121r9] clause 10.3	One (mandatory)

Table 31: (continued)

Names	Definition	Data type and values	Multiplicity and use
boundingBox BoundingBox	Bounding box surrounding selected dataset, in available CRS.	List of comma-separated real numbers, e.g. “minx1, minx2, maxx1, maxx2” or “minx1, minx2, minx3, maxx1, maxx2, maxx3”. (Any spaces used to separate values here are for presentation only, not normative.)	Zero or one (optional) Include when spatial selection is desired
spatialSelection SpatialSelection	Indicates method of selecting objects with BoundingBox	`String`, not empty. Value is one of “contains_center”, “overlaps”, “cut”. Default is “overlaps”.	Zero or one (optional)
layers Layers	List of layer to retrieve the data from	`StringList` type, comma separated list of layer Identifiers	One (mandatory)
styles Styles	List of service styles to be applied to the layers	`StringList`, comma-separated list of one rendering style per requested layer	Zero or one (optional)
background Background	Identifier of desired background	Character String type, not empty Values are specified in service metadata	Zero or one (optional) Include when background desired
lods LODs	List of LODs requested for the layers	`StringList`, comma separated list of `URIs`	Zero or one (optional)
lodSelection LODSelection	Indicates method for selecting LODs	`String`, Values are specified in service metadata	Zero or one (optional)
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	Names of the delivery options requested by the client	`StringList`, comma separated list of `URIs`	Zero or more (optional)
exceptions Exceptions	Format of exceptions	ows:MimeType, see [OGC 06-121r9] clause 10.5	Zero or one (optional)
offset Offset	Offset vector which shall be applied to the scene, using the mode specified (if any)	Character String type, list of coordinate value of length 2 or 3, separated by comma [b]. Optionally followed by a colon and an explicit offset mode name	Zero or one (optional)
format Format	Format encoding of the scene	ows:MimeType, see [OGC 06-121r9] clause 10.5	One (mandatory)
viewpoints Viewpoints	Add Viewpoints to choose from	Character String type, list of tuple value, separated by comma.	Zero or one (optional)

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` module 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 32 lists the image layer types suggested by the 3DPS `View` module. These image layers and suggested encodings are described in the following.

Table 32: Image layer names and descriptions

Names	URI	Description
color Color	http://www.opengis.net/spec/3DPS/1.0/-concept/service/view/imageLayers/color	contains color value for each pixel
depth 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 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 Normal	http://www.opengis.net/spec/3DPS/1.0/-concept/service/view/imageLayers/normal	encodes the surface normal for the surface represented
mask Mask	http://www.opengis.net/spec/3DPS/1.0/-concept/service/view/imageLayers/mask	encodes for each pixel, whether any feature is visible there

10.2.1.1 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 many applications. 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 feature identifiers. 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.

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.

10.2.1.2 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 33 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 33: 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 (inefficient due to 24 bit data storage)
	image/png [a]
	image/png;mode=1bit
[a] Server chooses bit depth.	

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 RGB for

24 bit encoding or RGBA for 32 bit encoding. Figure 10 shows an example of encoding depth values as 32 bit RGBA colors. 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.

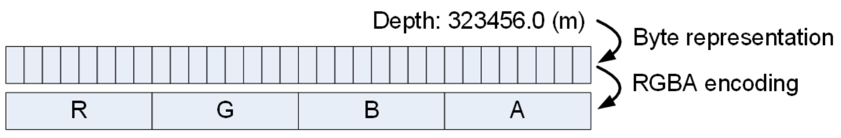

Figure 10: Example of encoding a depth value as 32 bit RGBA color components

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 RGB for 24 bit encoding or RGBA for 32 bit encoding. Figure 11 shows an example of encoding object-ids as 32 bit RGBA colors. 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.

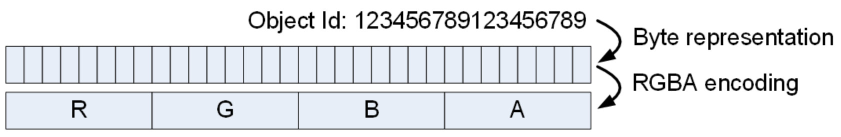

Figure 11: Example of encoding an object-id as 32 bit RGBA color components

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 (R,G,B) of a 24 bit color image. A surface normal shall be encoded as 24 RGB color values 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 lead to a normalized normal n = (x, y, z).
4. For each X, Y, and Z component of the transformed normal a decimal value is computed by adding 1 and dividing the result by 2.
5. These values represent the color components, R, G, and B respectively. Endianness shall be RGB.

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 12). 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, for example, projected spherical or cylindrical.

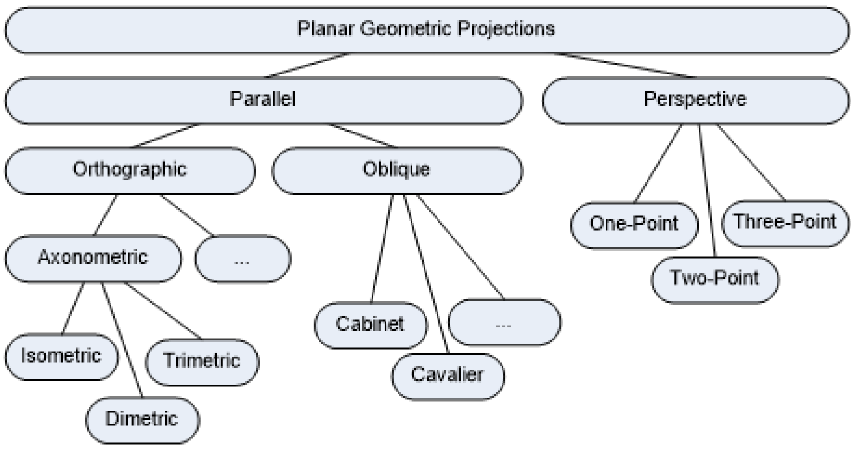

Figure 12: Examples of Projection types.

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 13).

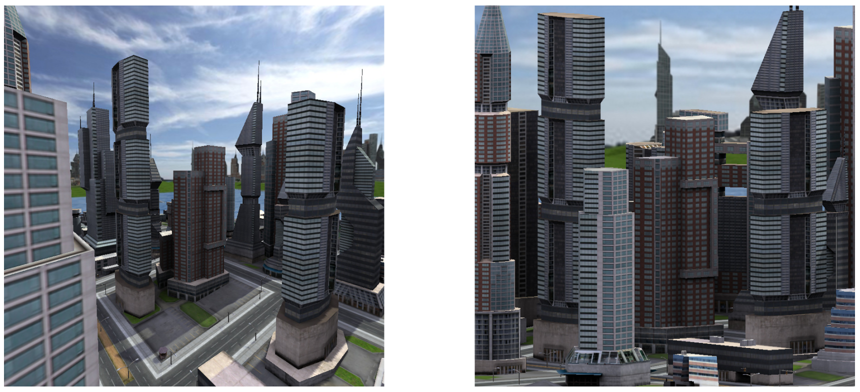

Figure 13: Examples of perspective (left) and orthographic (right) projections.

The 3DPS `View` Extension specifies two planar projection types, a perspective projection type and an orthographic projection type, PerspectiveProjection and OrthographicProjection, as defined in Table 35 and Table 36.

A specific 3DPS service can provide additional projection types. Supported projections shall be advertised in the service metadata document.

Table 34: Supported ProjectionTypes and meaning

Identifier	**Description**
perspective Perspective	Perspective projection as defined in Table 35
orthographic Orthographic	Orthographic projection as defined in Table 36

Possible values for the parameters of each projection type should be advertised in the service's metadata.

Table 35: view:PerspectiveProjection components

Name	Definition	Data type	Multiplicity and use
poc POC	Position of the virtual 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
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 near to the camera 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 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 CRS, measure unit is meter			

Table 36: view:OrthographicProjection components

Name	Definition	Data type	Multiplicity and use
poc POC	Position of the virtual 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
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

Table 36: (continued)

Name	Definition	Data type	Multiplicity and use
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 near to the camera 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 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 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 degree. FOVX specifies the horizontal angle of view, FOVY specifies the vertical angle of view. The service shall suggest default values for FOVX and FOVY. If FOVX and FOVY are not provided, the service shall use default values. If only one of FOVX or FOVY is provided, the service shall compute the missing value from the aspect ration of the requested Width and Height.

Left, Right, Bottom, Top In `OrthographicProjection` data structure, the optional Left, Right, Bottom, and Top parameter 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.

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 Image Coordinate System

The 3DPS `View` Extension uses two principal classes of Coordinate Systems (CS): an Image CS applicable to the image showing the 3D view generated by the 3DPS, and the Layer CRS used for specifying the data source BoundingBox.

An Image CS is a coordinate system 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 (e.g., GetView) 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).

10.2.4 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 service 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 Capabilities document.

Requirement 11: 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 the `Profile` element of the `ServiceIdentification` in a `GetCapabilities` request: 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 12: 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 service's OperationsMetadata 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

Requirement 13: http://www.opengis.net/spec/3DPS/1.0/req/service/view/layer-extension

A 3DPS service implementing conformance class *view* of this View Extension shall extend the core:Layer Extension element as defined in Figure 14 and Table 37
Dependency: 3DPS Core

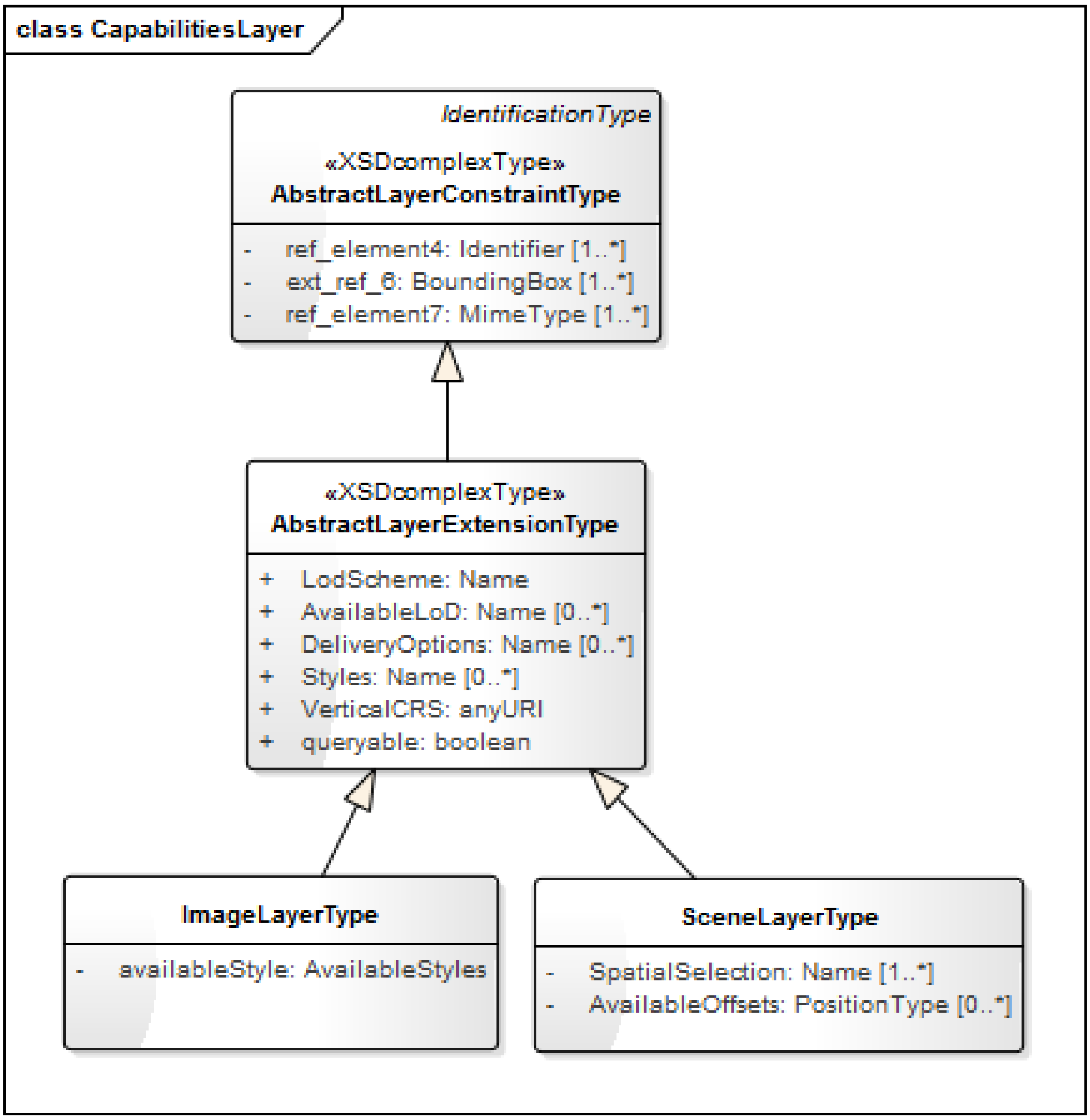
class CapabilitiesLayer
IdentificationType
«XSDcomplexType»
AbstractLayerConstraintType
- ref_element4: Identifier [1..*]
- ext_ref_6: BoundingBox [1..*]
- ref_element7: MimeType [1..*]
«XSDcomplexType»
AbstractLayerExtensionType
+ LodScheme: Name
+ AvailableLoD: Name [0..*]
+ DeliveryOptions: Name [0..*]
+ Styles: Name [0..*]
+ VerticalCRS: anyURI
+ queryable: boolean
ImageLayerType
- availableStyle: AvailableStyles
SceneLayerType
- SpatialSelection: Name [1..*]
- AvailableOffsets: PositionType [0..*]

Figure 14: 3DPS `view:Layer` extensions UML class diagram

Table 37: Extension of the `core:Layer` structure.

Names	Definition	Data type and values	Multiplicity and use
availableStyle AvailableStyle	Style available for this layer	Style data structure, see AvailableStyle	Zero or more (optional)

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* shall extend the core:PortrayalCapabilities Extension element as defined in Figure 15, Table 38, Table 39, Table 32, Table 40, and Table 41
Dependency: 3DPS Core (http://www.opengis.net/spec/3DPS/1.0/conf-class/core)

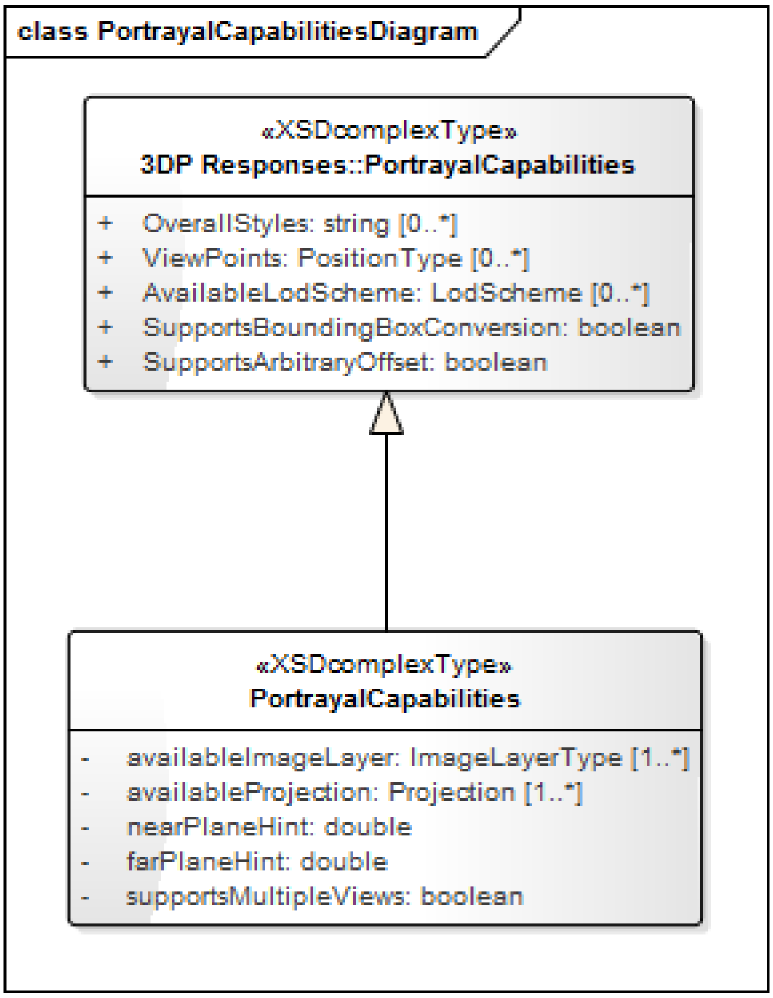
class PortrayalCapabilitiesDiagram
«XSDcomplexType»
3DP Responses::PortrayalCapabilities
+ OverallStyles: string [0..*]
+ ViewPoints: PositionType [0..*]
+ AvailableLodScheme: LodScheme [0..*]
+ SupportsBoundingBoxConversion: boolean
+ SupportsArbitraryOffset: boolean
«XSDcomplexType»
PortrayalCapabilities
- availableImageLayer: ImageLayerType [1..*]
- availableProjection: Projection [1..*]
- nearPlaneHint: double
- farPlaneHint: double
- supportsMultipleViews: boolean

Figure 15: 3DPS `view:PortrayalCapabilities` UML class diagram

Table 38: Extension of the `core:PortrayalCapabilities` structure.

Names	Definition	Data type and values	Multiplicity and use
availableImageLayer AvailableImageLayer	Image layers that can be served by this service	ImageLayer type, see Table 39 At least default "COLOR" layer shall be supported	One or more (mandatory) One for each supported ImageLayer
availableProjection AvailableProjection	A projection that is supported by this service	Projection type, see Table 40	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 views or image layers	Boolean type Default shall be "true"	Zero or one (optional) Include when multipart response is not supported

Table 39: Parts of ImageLayer data structure

Name	Definition	Data type	Multiplicity
identifier Identifier	Unambiguous identifier of this image layer, unique for this service	CodeList type, not empty	One (mandatory)
availableFormat AvailableFormat	Formats that are available for this ImageLayer	List of ows:MIME type, not empty, see [OGC 06-121r9] clause 10.5	One or more (mandatory) Include one for each supported image layer format

Table 40: Parts of Projection data structure

Name	Definition	Data type	Multiplicity
title Title	Title of this projection, human readable	`LanguageString`, see [OGC 06-121r9] clause 10.7	Zero or one (optional)
abstract Abstract	Brief narrative description of this projection	`LanguageString`, see [OGC 06-121r9] clause 10.7	Zero or one (optional)
identifier Identifier	Identifier of the projection type, unique for this service	Character Sting type, not empty	One (mandatory)
projectionParameter ProjectionParameter	Parameter that specifies this projection	DomainType, see [OGC 06-121r9] clause 13.2.1 This data type has an attribute described in Table 41	Zero or more (optional) Include one for each parameter of the projection

Table 41: Parts of ProjectionParameter data structure

Name	Definition	Data type	Multiplicity
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 other than default “true” desired

10.3.4.1 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 provide a hint on such values.

10.3.4.2 SupportsMultipleViews

This flag signals if a 3DPS service supports responding multiple views or multiple image layers by a single GetView response as HTTP multipart/mixed message (see Section 10.5.1).

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 16 and Table 42.

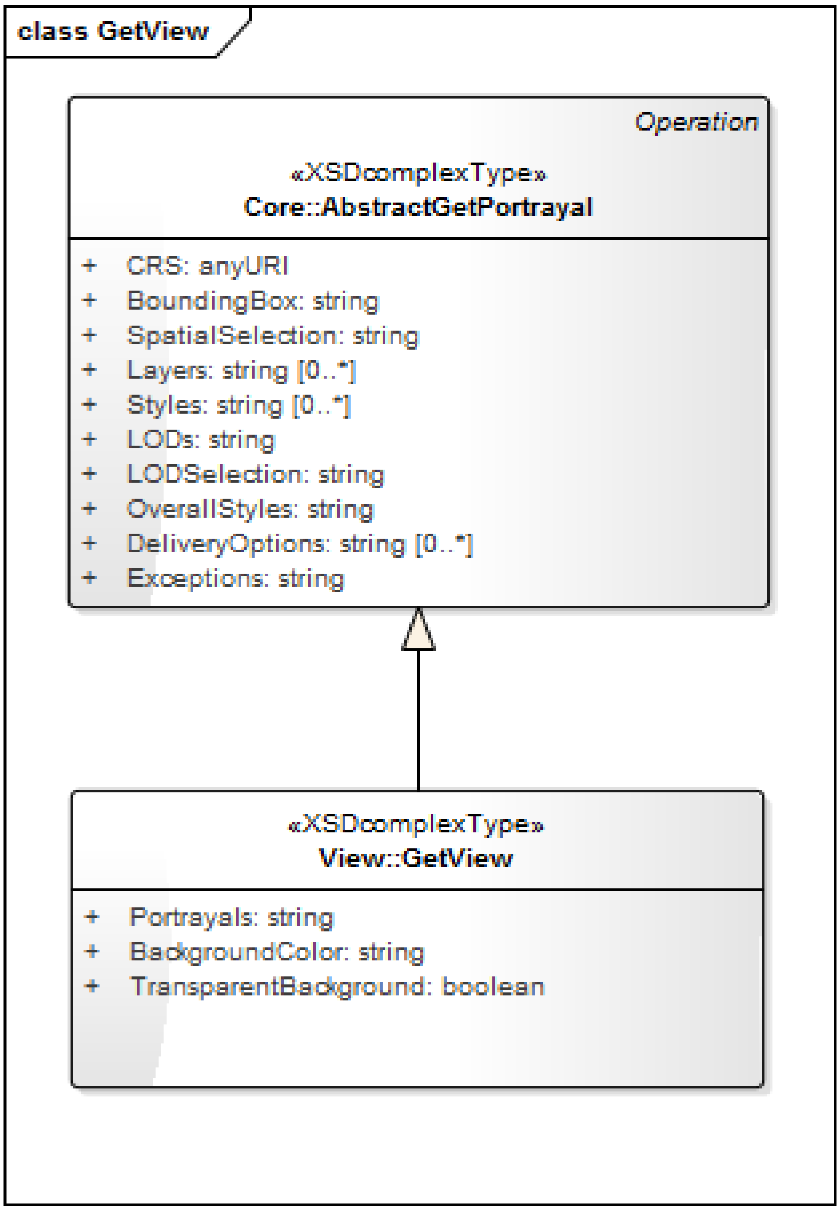

Figure 16: GetView request UML class diagram

Table 42: Additional properties of the GetView request

Names	**Definition**	**Data type and values**	**Multiplicity and use**
backgroundColor BackgroundColor	Background color desired	Character String type, not empty Hexadecimal RGB color values, format 0xRRGGBB. Default is 0xFFFFFF (white)	Zero or one (optional) Include when background color other than white desired

Table 42: (continued)

Names	Definition	Data type and values	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 PortrayalOutput type, not empty	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 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 Formats 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 Transparent parameter is set to “false”, non-data pixels shall be set to the value of BackgroundColor (see 7.3.3.10).

A 3DPS service shall generate a transparent background only, when no Background and BackgroundColor parameter 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 `PortrayalOuput` components, which specify image size (width and height), projections, output formats, and image qualities as defined in Table 43.

Table 43: Components of the `view:PortrayalOutput` data structure.

Names	Definition	Data type and values	Multiplicity and use
width Width	Width of desired output, in pixels	PositiveInteger type	One (mandatory)
height Height	Height of desired output, in pixels	PositiveInteger type	One (mandatory)
projections Projections	Camera specification and projection parameters	List of Projection data structure, see Table 35 and Table 36, not empty Supported projections are specified in service metadata	One (mandatory)
imageLayers ImageLayers	Reference(s) to image layers to retrieve	List of Character String type, not empty Supported ImageLayers (and formats) are specified in service metadata	One (mandatory)
formats Formats	Format encoding(s) of ImageLayer(s)	List of ows:MIME type, not empty, see [OGC 06-121r9] clause 10.5. List must have same length as in parameter ImageLayers Values are specified in service metadata	One (mandatory)
qualities Qualities	Integer that specifies desired quality of portrayal view (e.g. data resolution, rendering accuracy) [a]	List of Integer type, can be empty. List must have same length as lists in parameters Projections, ImageLayers, and Formats. List items can be empty Values from 0 to 100. Default is 100	Zero or one (optional) Include when desired

[a] The Quality parameter is only useful for formats supporting lossy compression. For other output formats the Quality parameter shall be ignored. The Quality 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.

10.4.3.1 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 Section 10.2.3) 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 Map 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.

10.4.3.2 Projections

The mandatory Projections parameter specifies a list of projections to apply for generating the 3D views. A projection contains the specification of the virtual camera and projection parameters as defined in Section 10.2.2, Table 35 and Table 36.

10.4.3.3 ImageLayers

The component ImageLayers contains a list of references to one or more ImageLayers advertised by the 3DPS as advertised in it’s service metadata, e.g., one of those listed in Table 32.

10.4.3.4 Formats

The mandatory Formats parameter specifies a list of image formats as ows:MimeType, see [OGC 06-121r9] clause 10.5. 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 39).

10.4.3.5 Qualities

The optional Qualities parameter specifies a list of integers, which describe for each requested ImageLayer and Format the visual quality of the output. Values are between 0 and 100. Empty values are possible and refer to the default value (100).

The Qualities parameter is only 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" (i.e., the character sequence comprised of 'i', 'n', 'i', 'm', 'a', 'g', 'e') 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 Format 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.

A service shall advertise the supported Exception values as AllowedValues of a Parameter element with name set to "Exception-Format" in the OperationsMetadata.

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 Format 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 Format parameter for each image layer.

10.5.1 MIME multipart response

If multiple 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.

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 in the Format parameter for this image layer. Each part contains a header that declares the MIME type of this message part.

Table Table 44 specifies the number and order of image layers in the GetView multipart response, if multiple Projections and/or multiple ImageLayers are requested in one PortrayalOutput. If multiple PortrayalOutputs are requested, the resulting image layer sets are concatenated in the same way, i.e., corresponding to their order in the GetView request.

Table 44: Number and order of responded image layers of one requested PortrayalOutput in GetView multipart/mixed response

Requested Projections	Requested ImageLayers	Number and order of result image layers
multiple (N)	one	Number=N Order is the same as that of requested Projections
one (1)	multiple (N)	Number=N Order is the same as that of requested ImageLayers
multiple (M)	multiple (N)	Number=M*N Order is: for each Projection (order as in request) all ImageLayers (order as in 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 shall 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 [OGC 06-121r9].

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 [OGC 06-121r9]. **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;
&Projections=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 45. The parameters listed in Table 45 shall be as specified in Table 42 and Table 43.

Table 45: GetView operation request URL parameters

Name [a]	Optionality and use	Definition and format	Example
service	Mandatory	Service type identifier	service=3DPS

Table 45: (continued)

Name [a]	**Optionality and use**	**Definition and format**	**Example**
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
layers Layers	Mandatory	Identifier(s) of desired data layer(s)	layers=dem,bldgs
styles 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 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

Table 45: (continued)

Name [a]	Optionality and use	Definition and format	Example
portrayals Portrayals	Mandatory	List of specifications of Width (one), Height (one), Projections (one or more), ImageLayers (one or more), Formats (one or more), Qualities (one or more)	Portrayals= WIDTH=1024;HEIGHT=1024; Projections= Perspective, 202000,3310000,200, 202000,3305000,200, 0,0,1, 60,, 0.01,1000.0; IMAGE-LAYERS=COLOR,DEPTH; FOR-MATS=image/jpeg,image/png%3Bmode=32bit; QUALITIES=100,100 @ WIDTH=768;HEIGHT=768; Projec-tions= Perspective, 202000,3310000,200, 202000,3305000,200, 0,0,1, 60,, 0.01,1000.0; IMAGELAYERS=COLOR,DEPTH; FOR-MATS=image/png,image/png; QUALI-TIES=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 [OGC 06-121r9]. [b] The value for this field shall be encoded as specified in section TODO

10.7.1.1 BoundingBox

The value of the BoundingBox parameter is a list of comma-separated real numbers as described [OGC 06-121r9] clause 10.2.3. The units, ordering, and direction of increment of the X and Y axes are as defined by the request's CRS parameter. A 3D BoundingBox is KVP encoded in the form "minx,miny,minz,maxx,maxy,maxz".

10.7.1.2 Layers

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

10.7.1.3 Styles

The mandatory Styles parameter shall be a comma-separated list of Style Identifiers as advertised in the service's metadata Contents section. The list shall contain one Style identifier for each Layer in the Layers parameter; the order shall be the same. Thus, the Style list shall have the same length as the Layer list. The Style list can be empty; in this case the default style shall be applied to all Layers. An entry of the Styles list can be empty; in this case the default Style of the corresponding Layer shall be applied.

10.7.1.4 KVP encoding of Portrayals

PortrayalOutput KVP encoding

Encoding of the PortrayalOutput value fields shall be as follows:

a. An at symbol (@) shall be used to separate one PortrayalOutput value from the next.

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

c. Missing mandatory PortrayalOutput parameter names shall raise a `MissingParameterValue` exception.

d. An equal sign (=) shall be used to separate a PortrayalOutput parameter name from its value.

e. All PortrayalOutput parameter values shall be encoded using the standard Internet practice for encoding URLs [RFC 1738].

The PortrayalOutput parameter's value, in a KVP GetView request, shall conform to the following grammar (using EBNF, Extended Backus-Naur Form notation [ISO/IEC 14977]):

```
Portrayals = PortrayalOutput, {"@", PortrayalOutput};
PortrayalOutput = "WIDTH=", 'image width',
    ";HEIGHT=", 'image height',
    ";PROJECTIONS=", ProjectionList,
    ";IMAGELAYERS=", ImageLayerList,
    ";FORMATS=", FormatList,
    ";QUALITIES=" QualityList;

ProjectionList = Projection, {",", Projection};
Projection = ProjectionTypeName, {",", ProjectionParameterValue};
ProjectionTypeName =    "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';
```

URL encoding (according to [RFC 3986]) is required for parameter values. For example the semicolon (;) within a parameterized MIME type identifier has to be escaped by "%3B":

EXAMPLE: The encoding for "image/png;mode=32Bit" is "image/png%3Bmode=32Bit".

Projection parameters KVP encoding

The Projections parameter (which is a parameter of the PortrayalOutput type) shall be encoded as comma separated list of tuple values. Multiple projection specifications are concatenated with a comma as separator. For each requested Projection, the size of the 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 35 and Table 36) the values can be empty. In this case, the service has to use default values, ignore parameters, or compute values. The Projections parameter shall contain multiple projections only, when the 3DPS service advertises the capability to generate multiple view and/or image layers element in the service metadata.

10.7.2 GetView request XML encoding (optional)

A 3DPS services may implement HTTP POST transfer of the GetView operation request using XML encoding. The following schema fragment specifies the contents and structure of a GetView operation request encoded in XML:

```
<!-- =========================================================== -->
<element name="GetView" type="view:GetViewType"/>
<!-- =========================================================== -->
<complexType name="GetViewType">
  <annotation>
    <documentation>XML encoded 3DPS GetView operation request.  </documentation>
  </annotation>
  <complexContent>
```

```
    <extension base="core:PSRequestBaseType">
      <sequence>
        <element name="CRS" type="anyURI"/>
        <element ref="ows:BoundingBox" minOccurs="0"/>
        <element name="SpatialSelection" type="ows:CodeType" minOccurs="0"/>
        <element name="Layers" type="core:IdentifierListType" minOccurs="1"/>
        <element name="Styles" type="core:IdentifierListType" minOccurs="0"/>
        <element name="OverallStyles" type="core:IdentifierListType" minOccurs="0"/>
        <element name="Background" type="string" minOccurs="0"/>
        <element name="BackgroundColor" type="string" minOccurs="0"/>
        <element name="TransparentBackground" type="boolean" minOccurs="0"/>
        <element name="Portrayals" type="view:PortrayalOutputListType"/>
        <element name="Exceptions" type="string" minOccurs="0"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

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></ows:Identifier>
  </core:Styles>
  <core:OverallStyles>
    <ows:Identifier>fog</ows:Identifier>
    <ows:Identifier>abstract</ows:Identifier>
  </core:OverallStyles>
  <core:Background>skybox</core:Background>
  <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:Projections>
        <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: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:Projections>
      <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>
```

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 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 3D Portrayal service 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 WebFeatureService for further information pertaining that feature. The GetFeatureInfo operation is only supported for those Layers for which the attribute queryable="true" has been defined or inherited. A client should not issue a GetFeatureInfo request for other layers. A 3D Portrayal Service shall respond with a properly formatted service exception response (code = OperationNotSupported) 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 data sets.

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 the `Profile` element of the `ServiceIdentification` in a `GetCapabilities` request: 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 one Operation element in the service's OperationsMetadata for each of the implemented GetFeatureInfo operation and set its "name" attribute to a value as defined in Table 46. **Dependency:** 3DPS Core (http://www.opengis.net/spec/3DPS/1.0/conf-class/core)

Table 46: 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 [OGC 06-121r9] clause 10.5. **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:

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

11.2.3 Additions to 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 Extension element as defined in Figure 17 and Table 47
Dependency: 3DPS Core (http://www.opengis.net/spec/3DPS/1.0/conf-class/core)

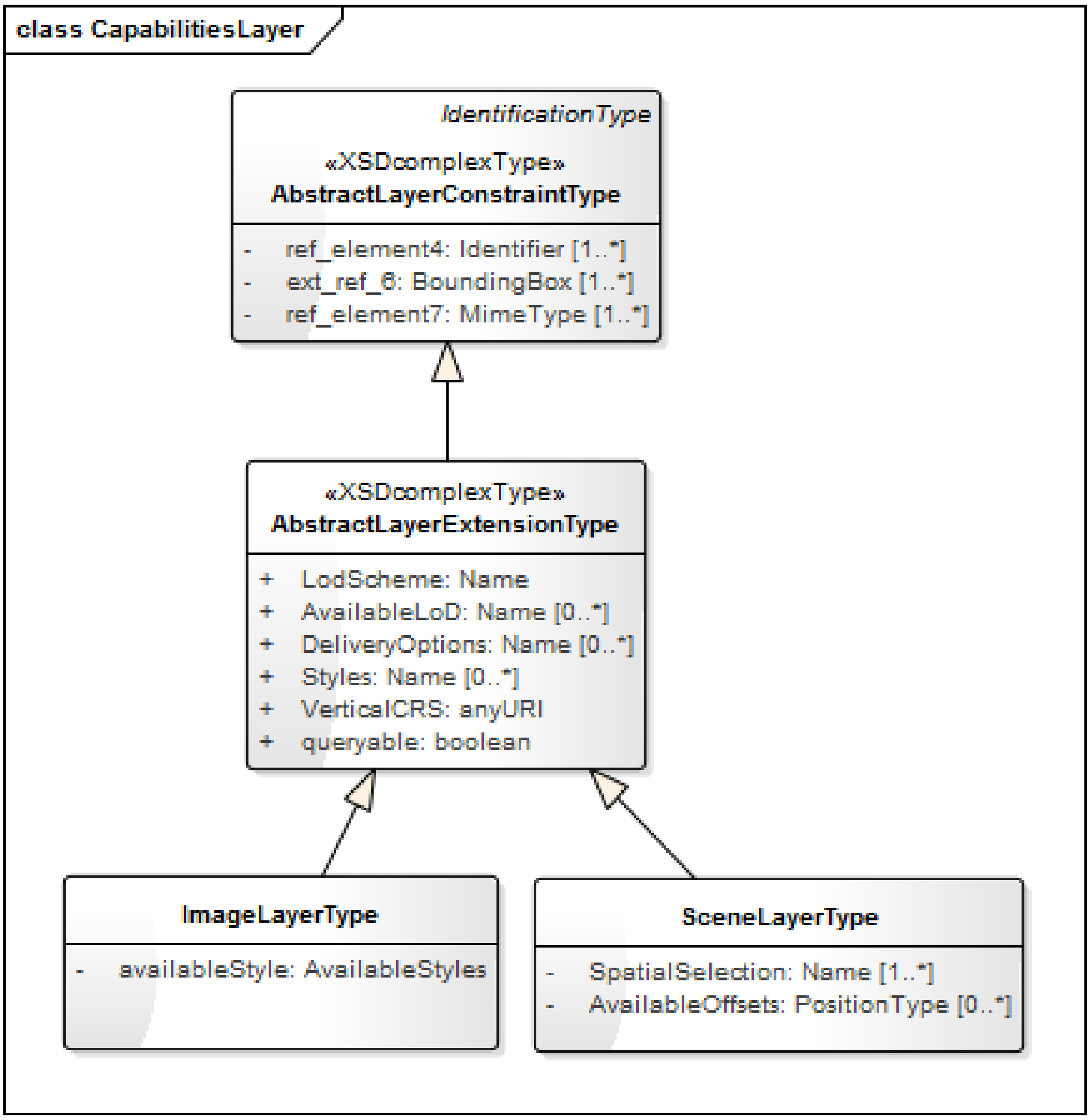
class CapabilitiesLayer
IdentificationType
«XSDcomplexType»
AbstractLayerConstraintType
- ref_element4: Identifier [1..*]
- ext_ref_6: BoundingBox [1..*]
- ref_element7: MimeType [1..*]
«XSDcomplexType»
AbstractLayerExtensionType
+ LodScheme: Name
+ AvailableLoD: Name [0..*]
+ DeliveryOptions: Name [0..*]
+ Styles: Name [0..*]
+ VerticalCRS: anyURI
+ queryable: boolean
ImageLayerType
- availableStyle: AvailableStyles
SceneLayerType
- SpatialSelection: Name [1..*]
- AvailableOffsets: PositionType [0..*]

Figure 17: 3DPS `info:Layer` extensions UML class diagram

In the info module, the Layer's description is enhaced with the queryable attribute.

Table 47: Extension of the `core:Layer` structure.

Names	**Definition**	**Data type and values**	**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 Abstract GetFeatureInfo 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 18 and Table 48.

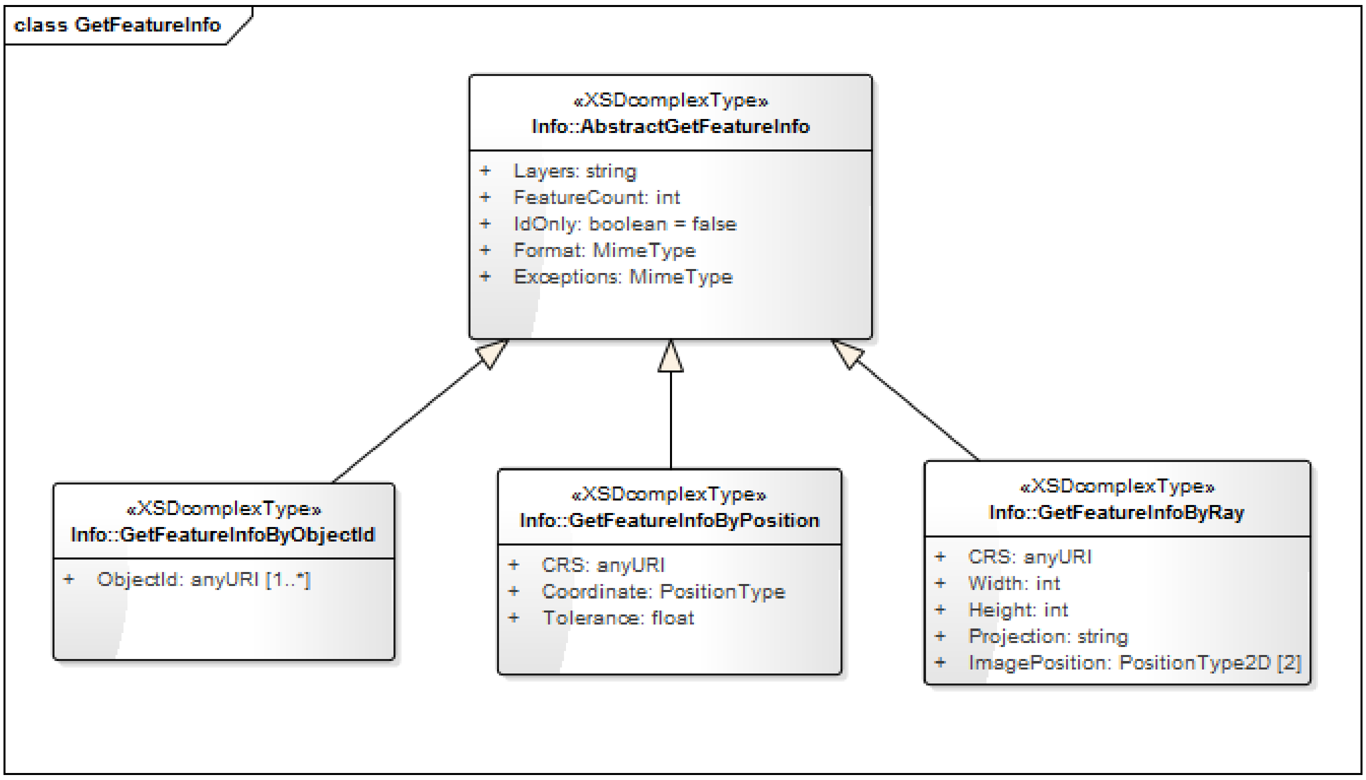

Figure 18: 3DPS `info:AbstractGetFeatureInfo` operation request UML class diagram

Table 48: Parameters of the AbstractGetFeatureInfo operation.

Names[a]	Definition	Data type and values	Multiplicity and use
layers Layers	List of layer to retrieve the data from	Character String type, comma separated list of layer Identifiers	One (mandatory)
featureCount FeatureCount	Number of features to return information from.	Integer 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 [OGC 06-121r9] clause 10.5	One (mandatory)

Table 48: (continued)

Names[a]	Definition	Data type and values	Multiplicity and use
exceptions Exceptions	Reference to format in which operation exceptions should be returned	ows:MimeType, see [OGC 06-121r9] clause 10.5	Zero or one (optional)

11.3.1 Layers

The Layers parameter specifies a comma separated 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 comma separated list shall refer to a layer identifier as described in the service's meta data.

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 (IdOnly = true). The feature identifier can be used to request further information of the feature from a WebFeatureService, for example. Default is IdOnly=false.

11.3.4 Format

The mandatory Format parameter specifies the target encoding of the returned attribute information provided as ows:MimeType, see [OGC 06-121r9] clause 10.5. Available formats are described in the service's meta data.

11.3.5 Exceptions

The optional Exceptions parameter specifies the behaviour 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's meta data parameter value, see 7.3.2.2.

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 18, Table 49 and Table 48.

Table 49: Parameters of the GetFeatureInfoByRay request.

Names	Definition	Data type and values	Multiplicity and use
crs CRS	CRS to apply to BoundingBox and Projection parameters	URI	One (mandatory)
width Width	Width of output image that the request refers to, in pixels	PositiveInteger type	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	Projection type, see Table TODO and Table TODO 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 mandatory Width and Height parameters specify the size in integer pixels of the view that describes the ray cast.

11.4.2 Projection

The mandatory 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 Section 10.2.2, Table 35 and Table 36.

11.4.3 ImagePosition

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

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.

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 18, Table 50 and Table 48.

Table 50: Parameters of the GetFeatureInfoByPosition operation.

Names[a]	**Definition**	**Data type and values**	**Multiplicity and use**
crs CRS	CRS of the coordinates	URI	One (mandatory)
coordinate Coordinate	Location coordinates used to search for features	Element of type Position3D. 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 parameter value for the coordinate reference system (CRS) is defined in [OGC 06-121r9] clause 10.3 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 mandatory 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 it provisional only, i.e. the service may return features ouside that range even when specified, and it also may use an implemetation-defined default if not specified.

11.6 GetFeatureInfoByObjectId request

The GetFeatureInfoByRay 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.

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.

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 18, Table 51 and Table 48.

Table 51: Parameters of the GetFeatureInfoByPosition operation.

Names	Definition	Data type and values	Multiplicity and use
objectId ObjectId	unique identifier of the selected feature(s)	URI	One or more (mandatory) featureCount FeatureCount

11.6.1 ObjectID

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

11.6.2 FeaureCount

If the featureCount is specified, it shall correspond to the number of object identifiers provided. Otherwise, featureCount shall be assumed to equal the cardinality of `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 52, Table 53, Table 54 and Table 55.

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 52: FeatureInfo components

Names	Definition	Data type and values	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 53: FeatureInfoList components

Names	Definition	Data type and values	Multiplicity and use
featureId FeatureId	Feature identifier that may be used to directly request an underlying feature database (e.g. OGC WFS)	Character String type, not empty	Zero or one (optional) Include if feature identifier is available
objectId ObjectId	Object identifier, local the the service instance	Character String type, not empty	Zero or one (optional) Include if object identifier is available
typeName TypeName	Name of the feature's type	Character String type, not empty	Zero or one (optional) + Include if a type name is available
featureAttributeList FeatureAttributeList	Group of feature attributes	FeatureAttributeList data type, not empty	Zero or more (optional) Include when attributes available for this feature

Table 54: FeatureAttributeList components

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

Table 55: FeatureAttribute components

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

11.7.1 FeatureInfo encoding

11.7.1.1 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>
```

11.7.1.2 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 3DPortrayalService service encounters an error while performing a GetFeatureInfo operation, it shall return an exception report message as specified in Clause 8 of [OGC 06-121r9]. The allowed standard exception codes shall include those listed in Table 56. For each listed exceptionCode, the contents of the “locator” parameter value shall be filled as specified in the right column of the table.

Note
To reduce the need for readers to refer to other documents, the first four values listed below are copied from Table 27 in Subclause 8.3 of [OGC 06-121r9].

11.7.3 Exception codes for GetFeatureInfo operation

Table 56: Exception codes of the GetFeatureInfo operation.

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
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

Table 56: (continued)

exceptionCode value	Meaning of code	“locator” value
NoApplicableCode	No other exceptionCode specified by this service and service applies to this exception	None, omit “locator” parameter

12 Annex A: Conformance Class Abstract Test Suite (Normative)

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 id	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 Response	
Requirement 1	
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 accoring to the declared schema and that the root element conforms to `core:CapabilitiesType`.

GetResource Request	
Requirement 2	
Test identifier	http://www.opengis.net/spec/3DPS/1.0/req/service/core/getResourceById/request
Test purpose	Test that the GetCapabilities response conforms to the specification.
Test method	Verify the services `core:GetResourceById` against [OGC 06-121r9]

getResourceId response	
Requirement 3	
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 innner stucture and the service responses mime type to match the requested MIME.’

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 id	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 `http://www.opengis.net/spec/3DPS/1.0/testing/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 4	
Test identifier	http://www.opengis.net/spec/3DPS/1.0/req/service/scene/extension-identifier
Test purpose	Test that an instance declares its implemetation of the scene requirements class.
Test method	Verify that the GetCapabilities answer includes the declaration of a profile element `http://www.opengis.net/spec/3DPS/1.0/extension/scene/1.0`

Test for Operations Metadata	
Requirement 5	
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 operantion named "GetScene". Verify that the operation declares the parameters given in Table 18 and Table 29 are declared.

Layer extension test	
Requirement 6	
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 there are layers which are declared with XML conforming to the `scene:SceneLayerType` schema

Mime types	
Requirement 7	
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 8	
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 element conform to the scene:PortrayalCapabilities schema.

Response	
Requirement 10	
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 id	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 `http://www.opengis.net/spec/3DPS/1.0/testing/conf-class/core`.

12.6 Tests for requirements of the view requirements class

Modifications to service identification	
Requirement 11	
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 answer includes the declaration of a profile element `http://www.opengis.net/spec/3DPS/1.0/extension/view/1.0`

Test for Operations Metadata	
Requirement 12	
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 18 and Table 42 are declared.

Layer extension test	
Requirement 13	
Test identifier	http://www.opengis.net/spec/3DPS/1.0/req/service/view/layer-extension
Test purpose	Ensure that the layer extensions for view-based portrayal are declared.
Test method	Verify that there are layers which are declared with XML conforming to the `view:ViewLayerType` schema

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 element conform to the core:PortrayalCapabilities schema.

Test exception use in response	
Requirement 16	
Test identifier	http://www.opengis.net/spec/3DPS/1.0/req/service/view/exception/common
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 id	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 `http://www.opengis.net/spec/3DPS/1.0/testing/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 implemetation of the info requirements class.
Test method	Verify that the GetCapabilities answer includes the declaration of a profile element `http://www.opengis.net/spec/3DPS/1.0/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
Test method	Verify that the GetCapabilities response includes at least one operation mentioned in Table 46.

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 purpose	Test that feature information formats are declared
Test method	Verify that each operation from Table 46 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 is declared conforming to info:Layer.

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 geature info operations extend AbstractGetFeatureInfo
Test method	Verify that each operation from Table 46 declared in the operations metadata also declares the parameters listed in Table 48.

Test GetFeatureInfoByRay request structure
Requirement 22

Test GetFeatureInfoByRay request structure	
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 18, Table 49 and Table 48. Verify that all parameters are declared.

Test GetFeatureInfoByRay 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 18, Table 50 and Table 48. Verify that all parameters are declared.

Test GetFeatureInfoByRay 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 18, Table 51 and Table 48. 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 feaure 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.

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

The schema content is being repeated here as a convenience to the reader. The schemas on `schema.opengis.org` are considered normative.

13.1 core schema

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema targetNamespace="http://www.opengis.net/3dps/1.0/core" xmlns="http://www.opengis ↩
    .net/3dps/1.0/core" xmlns:core="http://www.opengis.net/3dps/1.0/core" xmlns:view="http: ↩
    //www.opengis.net/3dps/1.0/view" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:ows=" ↩
    http://www.opengis.net/ows/2.0">
        <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/view" schemaLocation="3dps- ↩
            view.xsd"/>
        <xs:import namespace="http://www.opengis.net/ows/2.0" schemaLocation="http:// ↩
            schemas.opengis.net/ows/2.0/owsGetCapabilities.xsd"/>
        <xs:import namespace="http://www.opengis.net/ows/2.0" schemaLocation="http:// ↩
            schemas.opengis.net/ows/2.0/owsDataIdentification.xsd"/>
        <xs:complexType name="CapabilitiesType">
                <xs:complexContent>
                        <xs:extension base="ows:CapabilitiesBaseType">
                                <xs:sequence>
                                        <xs:element name="Portrayal" type=" ↩
                                            PortrayalCapabilities" minOccurs="1" maxOccurs=" ↩
                                            1"/>
                                </xs:sequence>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="AbstractLayerConstraintType">
                <xs:complexContent>
                        <xs:extension base="ows:IdentificationType">
                                <xs:sequence>
                                        <xs:element ref="ows:Identifier" minOccurs="1"  ↩
                                            maxOccurs="unbounded"/>
                                        <xs:element ref="ows:BoundingBox" minOccurs="1"  ↩
                                            maxOccurs="unbounded"/>
                                        <xs:element name="ref_element7" type="ows:MimeType" ↩
                                             minOccurs="1" maxOccurs="unbounded"/>
                                </xs:sequence>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="DeliveryOption">
                <xs:annotation>
                        <xs:documentation>Describe a delivery option, which is a service  ↩
                            usage convention targetting specific use cases, for example  ↩
                            streaming large scenes or optimizing for low bandwidth. Delivery ↩
                             options may be restricted to certain formats and layers.
```

```
A delivery option is understood to be a special mode to be invoked on top of the basic  ←
    delivery of data or geometry information. It may require special client capabilities.</ ←
    xs:documentation>
                </xs:annotation>
                <xs:sequence>
                        <xs:element name="name" type="xs:Name" minOccurs="1" maxOccurs="1">
                                <xs:annotation>
                                        <xs:documentation>The name used to refer to the  ←
                                            delivery option</xs:documentation>
                                </xs:annotation>
                        </xs:element>
                        <xs:element name="identifier" type="ows:CodeType" minOccurs="0"  ←
                            maxOccurs="1">
                                <xs:annotation>
                                        <xs:documentation>A code representing the delivery  ←
                                            option supported by this server. Each method may ←
                                             be freely defined in an own codespace, and the  ←
                                            standard documents a list of those which are  ←
                                            well-understood and deemed likely to foster  ←
                                            interoperability across service instances.</ ←
                                            xs:documentation>
                                </xs:annotation>
                        </xs:element>
                        <xs:element name="formats" type="ows:MimeType" minOccurs="0"  ←
                            maxOccurs="unbounded">
                                <xs:annotation>
                                        <xs:documentation>The formats which support this  ←
                                            delivery option. Empty means no restriction, all ←
                                             service formats support the option.</ ←
                                            xs:documentation>
                                </xs:annotation>
                        </xs:element>
                </xs:sequence>
        </xs:complexType>
        <xs:element name="Capabilities" type="core:CapabilitiesType"/>
        <xs:complexType name="PortrayalCapabilities">
                <xs:annotation>
                        <xs:documentation>This element contains a summary of capabilities  ←
                            provided by the service. It is intended to be useful for  ←
                            selecting capabilities to use in a concrete scenario, but there  ←
                            is no guarantee that any particular combination of them will  ←
                            work as intended.</xs:documentation>
                </xs:annotation>
                <xs:sequence>
                        <xs:element name="OverallStyles" type="xs:string" minOccurs="0"  ←
                            maxOccurs="unbounded">
                                <xs:annotation>
                                        <xs:documentation>A list of overall styles accepted ←
                                             by the server</xs:documentation>
                                </xs:annotation>
                        </xs:element>
                        <xs:element name="ViewPoints" type="ows:PositionType" minOccurs="0" ←
                             maxOccurs="unbounded">
                                <xs:annotation>
                                        <xs:documentation>A list of view points that might  ←
                                            be relevant to the client</xs:documentation>
                                </xs:annotation>
                        </xs:element>
                        <xs:element name="AvailableLodScheme" type="view:LodScheme"  ←
                            minOccurs="0" maxOccurs="unbounded">
                                <xs:annotation>
```

```
                                        <xs:documentation>The LoD schemes supported by the ↩
                                            server. The individiual level-of-detail item ↩
                                            names should be distinct, although typically ↩
                                            every layer would only support a single scheme.< ↩
                                            /xs:documentation>
                                </xs:annotation>
                        </xs:element>
                        <xs:element name="SupportsBoundingBoxConversion" type="xs:boolean" ↩
                            minOccurs="1" maxOccurs="1"/>
                        <xs:element name="SupportsArbitraryOffset" type="xs:boolean" ↩
                            minOccurs="1" maxOccurs="1"/>
                        <xs:element name="CapabilitiesType" type="CapabilitiesType" ↩
                            minOccurs="1" maxOccurs="1"/>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType name="AbstractLayerExtensionType">
                <xs:complexContent>
                        <xs:extension base="AbstractLayerConstraintType">
                                <xs:sequence>
                                        <xs:element name="LodScheme" type="xs:Name" ↩
                                            minOccurs="1" maxOccurs="1">
                                                <xs:annotation>
                                                        <xs:documentation>The name of the ↩
                                                            LoD scheme this layer supports</ ↩
                                                            xs:documentation>
                                                </xs:annotation>
                                        </xs:element>
                                        <xs:element name="AvailableLoD" type="xs:Name" ↩
                                            minOccurs="0" maxOccurs="unbounded">
                                                <xs:annotation>
                                                        <xs:documentation>List of level-of- ↩
                                                            detail values that are exected ↩
                                                            to hold data for that layer. ↩
                                                            They should be specified in the ↩
                                                            order they are specified in the ↩
                                                            LodScheme. Empty indicates that ↩
                                                            no level of detail processing is ↩
                                                             supported.</xs:documentation>
                                                </xs:annotation>
                                        </xs:element>
                                        <xs:element name="DeliveryOptions" type="xs:Name" ↩
                                            minOccurs="0" maxOccurs="unbounded">
                                                <xs:annotation>
                                                        <xs:documentation>the delivery ↩
                                                            options available for the layer. ↩
                                                             Empty means none are available. ↩
                                                            </xs:documentation>
                                                </xs:annotation>
                                        </xs:element>
                                        <xs:element name="Styles" type="xs:Name" minOccurs= ↩
                                            "0" maxOccurs="unbounded"/>
                                        <xs:element name="VerticalCRS" type="xs:anyURI" ↩
                                            minOccurs="1" maxOccurs="1">
                                                <xs:annotation>
                                                        <xs:documentation>The Vertical CRS, ↩
                                                             if known.</xs:documentation>
                                                </xs:annotation>
                                        </xs:element>
                                        <xs:element name="queryable" type="xs:boolean" ↩
                                            minOccurs="1" maxOccurs="1"/>
                                </xs:sequence>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="LevelOfDetail">
                <xs:sequence>
                        <xs:element name="Ordinal" type="xs:int" minOccurs="1" maxOccurs="1 ↩
                            "/>
                        <xs:element name="LodScheme" type="view:LodScheme" minOccurs="1"  ↩
                            maxOccurs="1"/>
                </xs:sequence>
        </xs:complexType>
</xs:schema>
```

13.2 scene schema

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema targetNamespace="http://www.opengis.net/3dps/1.0/scene" xmlns="http://www. ↩
    opengis.net/3dps/1.0/scene" xmlns:scene="http://www.opengis.net/3dps/1.0/scene"  ↩
    xmlns:core="http://www.opengis.net/3dps/1.0/core" xmlns:xs="http://www.w3.org/2001/ ↩
    XMLSchema" xmlns:ows="http://www.opengis.net/ows/2.0" xmlns:view="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="3dps- ↩
            core.xsd"/>
        <xs:complexType name="SceneLayerType">
                <xs:complexContent>
                        <xs:extension base="core:AbstractLayerExtensionType">
                                <xs:sequence>
                                        <xs:element ref="xs:Name" minOccurs="1" maxOccurs=" ↩
                                            unbounded"/>
                                        <xs:element ref="ows:PositionType" minOccurs="0"  ↩
                                            maxOccurs="unbounded"/>
                                </xs:sequence>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="PortrayalCapabilities">
                <xs:complexContent>
                        <xs:extension base="core:PortrayalCapabilities">
                                <xs:sequence>
                                        <xs:element ref="boolean" minOccurs="1" maxOccurs=" ↩
                                            1"/>
                                </xs:sequence>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
</xs:schema>
```

13.3 view schema

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema targetNamespace="http://www.opengis.net/3dps/1.0/view" xmlns="http://www.opengis ↩
    .net/3dps/1.0/view" xmlns:view="http://www.opengis.net/3dps/1.0/view" xmlns:core="http: ↩
    //www.opengis.net/3dps/1.0/core" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:ows=" ↩
    http://www.opengis.net/ows/2.0">
        <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/3dps/1.0/core" schemaLocation="3dps- ↩
    core.xsd"/>
<xs:complexType name="ImageLayerType">
        <xs:complexContent>
                <xs:extension base="core:AbstractLayerExtensionType">
                        <xs:sequence>
                                <xs:element ref="AvailableStyles" minOccurs="1"  ↩
                                    maxOccurs="1"/>
                                <xs:element name="LodScheme">
                                        <xs:complexType>
                                                <xs:annotation>
                                                        <xs:documentation>Describes ↩
                                                             a LoD scheme as used  ↩
                                                            for portrayal. The name  ↩
                                                            might refer to the  ↩
                                                            standard document or be  ↩
                                                            chosen arbitrarily.</ ↩
                                                            xs:documentation>
                                                </xs:annotation>
                                                <xs:sequence>
                                                        <xs:element ref="xs:Name"  ↩
                                                            minOccurs="1" maxOccurs= ↩
                                                            "1">
                                                                <xs:annotation>
                                                                        < ↩
                                                                            xs:documentation
                                                                            >The  ↩
                                                                            name  ↩
                                                                            used to  ↩
                                                                            refer to ↩
                                                                             the  ↩
                                                                            level-of ↩
                                                                            -detail  ↩
                                                                            scheme.  ↩
                                                                            The name ↩
                                                                             is  ↩
                                                                            intended ↩
                                                                             to be  ↩
                                                                            valid as ↩
                                                                             a name  ↩
                                                                            local to ↩
                                                                             the  ↩
                                                                            service  ↩
                                                                            instance ↩
                                                                            , while  ↩
                                                                            the  ↩
                                                                            identifier ↩
                                                                              ↩
                                                                            provides ↩
                                                                             a  ↩
                                                                            global  ↩
                                                                            complement ↩
                                                                            .</ ↩
                                                                            xs:documentation
                                                                            >
                                                                </xs:annotation>
                                                        </xs:element>
                                                        <xs:element ref=" ↩
                                                            ows:CodeType" minOccurs= ↩
                                                            "0" maxOccurs="unbounded ↩
                                                            ">
                                                                <xs:annotation>
```

```
                                                                        < ↩
                                                                           xs:documentation
                                                                           >To  ↩
                                                                           unambiguously ↩
                                                                             ↩
                                                                           identify ↩
                                                                            the  ↩
                                                                           scheme</ ↩
                                                                           xs:documentation
                                                                           >
                                                                </xs:annotation>
                                                        </xs:element>
                                                        <xs:element ref=" ↩
                                                            core:LevelOfDetail"  ↩
                                                            minOccurs="1" maxOccurs= ↩
                                                            "unbounded">
                                                                <xs:annotation>
                                                                        < ↩
                                                                           xs:documentation
                                                                           >Ordered ↩
                                                                            list of ↩
                                                                            names,  ↩
                                                                           each  ↩
                                                                           referring ↩
                                                                            to a  ↩
                                                                           successively ↩
                                                                            lower  ↩
                                                                           level-of ↩
                                                                           -detail< ↩
                                                                           / ↩
                                                                           xs:documentation
                                                                           >
                                                                </xs:annotation>
                                                        </xs:element>
                                                </xs:sequence>
                                        </xs:complexType>
                                </xs:element>
                        </xs:sequence>
                </xs:extension>
        </xs:complexContent>
</xs:complexType>
<xs:complexType name="PortrayalCapabilities">
        <xs:complexContent>
                <xs:extension base="core:PortrayalCapabilities">
                        <xs:sequence>
                                <xs:element ref="ImageLayerType" minOccurs="1"  ↩
                                    maxOccurs="unbounded"/>
                                <xs:element ref="Projection" minOccurs="1"  ↩
                                    maxOccurs="unbounded"/>
                                <xs:element ref="double" minOccurs="1" maxOccurs="1 ↩
                                    ">
                                        <xs:annotation>
                                                <xs:documentation>must not be  ↩
                                                    smaller zero</xs:documentation>
                                        </xs:annotation>
                                </xs:element>
                                <xs:element ref="double" minOccurs="1" maxOccurs="1 ↩
                                    "/>
                                <xs:element ref="boolean" minOccurs="1" maxOccurs=" ↩
                                    1"/>
                        </xs:sequence>
                </xs:extension>
                </xs:complexContent>
        </xs:complexType>
</xs:schema>
```

These XML Schema Documents use (i.e. import) and build on the OWS common XML Schema Documents specified in [OGC 06-121r9], named:

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

All these XML Schema Documents contain documentation of the meaning of each element and attribute, and this documentation shall be considered normative as specified in Subclause 11.6.3 of [OGC 06-121r9].

14 Annex C (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 (DeliveryOptions) which shall be used to avoid exposing uncommon practises to clients which are not prepared to handle them.

15 Annex D (non-normative)

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:

```
PROFILE Interactive
COMPONENT Geospatial
```

Table 57: 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., and Leclerc, Y.G. 2000. Under the hood of GeoVRML 1.0. Proceedings of the fifth symposium on Virtual reality modeling language (Web3D-VRML) 23-28.

[2] McCann, M., Puk, R., Hudson, A., Melton, R., and Brutzman, D. 2009. Proposed enhancements to the x3d geospatial component. Proceedings of the 14th International Conference on 3D Web Technology 155-158.

[3] Yoo, B., and Brutzman, D. 2009. X3D earth terrain-tile production chain for georeferenced simulation. Proceedings of the 14th international conference on 3D Web technology 159-166.

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

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

[6] Reitz, T., Krämer, M., and Thum, S. 2009. A processing pipeline for X3D earth-based spatial data view services. Proceedings of the 14th international conference on 3D web technology 137-145

16 Annex E: Revision History

Table 58: 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 TC
07/2015	-	Simon Thum Benjamin Hagedorn	all	prepare for OAB review