Open Geospatial Consortium Inc.

Date: 2010-02-01

Reference number of this document: OGC 09-166r2

Version: 0.3.0

Category: OpenGIS® Discussion Paper

Editor: Benjamin Hagedorn

Web View Service Discussion Paper

Warning

This document is not an OGC Standard. This document is an OGC Discussion Paper and is therefore not an official position of the OGC membership. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an OGC Standard. Further, an OGC Discussion Paper should not be referenced as required or mandatory technology in procurements.

Document type:	OpenGIS® Discussion Paper
Document subtype:	Candidate Standard
Document stage:	Draft – for discussion only
Document language:	English

Contents

Page

Figures

Page

Tables

Page

i. Preface

The previous version of this document is the draft “Web Perspective View Service (WPVS) Implementation Specification” [7]. WPVS was the result of the work by the Web Terrain Service (WTS) Revision Working Group. This version of the Web View Service contains significant modifications and extensions to previous versions, both WPVS and WTS.

Although this draft standard defines a stand-alone web service for the 3D visualization of geographic data, the document refers to other OGC standards. It is based on the OWS Common Implementation Standard [OGC 06-121r3]. Concepts and definitions are derived from other OGC Implementation Standards.

Suggested additions, changes, and comments on this draft report are welcome and encouraged. Such suggestions may be submitted by email message or by making suggested changes in an edited copy of this document.

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

ii. Document terms and definitions

This document uses the standard terms defined in Subclause 5.3 of [06-121r3], which is based on the ISO/IEC Directives, Part 2. Rules for the structure and drafting of International Standards. In particular, the word “shall” (not “must”) is the verb form used to indicate a requirement to be strictly followed to conform to this standard.

iii. Document contributor contact points

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

Name	Organization
Benjamin Hagedorn	Hasso-Plattner-Institute at the University of Potsdam
Dieter Hildebrandt	Hasso-Plattner-Institute at the University of Potsdam
Jürgen Döllner	Hasso-Plattner-Institute at the University of Potsdam

iv. Revision history

Date	Release	Editor	Primary clauses modified	Description
2001-05-05	0.1.0	Raj Sing		MPP-1 (Military Pilot Project Initiative, Phase 1) Internal document
2001-06-27	0.2.0	Raj Singh		MPP-1 Development document
2001-06-11	0.3.0	Raj Singh		MPP-1 Draft
	0.3.1			Typographic corrections
2001-08-24	0.3.2	Raj Singh		OGC Interoperability Program Report (IPR): Web Terrain Server (WTS)
2002-08-19	0.3.3	Joshua Lieberman		Incorporating comments from IPR
2003-09-22	0.5.0	Joshua Lieberman, Jerome Sonnet		OGC Internal Draft for RFC
2005-10-20	n.n.n	Arliss Whiteside		Revised version: Web Perspective View Service (WPVS), internally known as Version 1.0.0
2009-06-30	0.3.0	Benjamin Hagedorn	all	Web View Service Draft: alignment with Web 3D Service Draft, adding non color-layers, adding GetFeatureInfo operation adding GetPosition operation, adding GetMeasurement operation, adding GetCamera operation, aligning with new W3DS draft, edits for OGC conformance.
2010-01-18	0.3.0	Carl Reed	Various	Prepare for publication as a DP

v. Changes to the OGC Abstract Specification

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

vi. Future work

Improvements in this document are desirable to improve the acceptance in the industry and scientific community.

Extending the data layer concept

So far, aligned to the draft W3DS version 0.4.0 [9], the WVS defines a data layer as a set of features. It might be worth, to A) classify layers, e.g., as a terrain layer and B) to consider also maps and coverages as data layers, which, e.g., could be draped over a terrain.

Styling

The current OGC Symbology Encoding (SE) [OGC 05-077r4] standard is not usable for 3D graphics and 3D geovirtual environments. Extensions for describing materials, surfaces, 3D transformations and so forth are needed and currently researched [OGC 09-042]. — This version of the WVS specification considers user-defined styling by the OGC Style Layer Descriptor (SLD) encoding standard. However, mechanisms need to be reviewed and maybe extracted into another document, such as in the case of the OGC Web Map Service interface standard (WMS), SE, and SLD for WMS.

Picking thematic data

In 3D geovirtual environments, thematic data can be integrated and encoded as appearance parameters of the contained features. For example, thermal images could be draped on the corresponding buildings façades. A WVS server could be used for visualizing this data. However, so far there are no capabilities specified to directly access the thematic information, e.g., simply clicking on the building façade.

SOAP interface

A SOAP interface may be desirable, but it is not yet specified in the OWS Common Implementation Standard. As soon as the OGC decides on a common framework for adding SOAP interfaces, this may be incorporates in this document.

Tiling Mechanism

As provided by Tiled Web Map Service [5] or the scene graph-based Web 3D Service [9], a tiling mechanism could be useful for the image-base 3D portrayal service, too. A tiled WVS could provide easy access to 3D geovisualization for tile-based client applications.

Foreword

This version of the draft standard replaces all previous versions.

This document is based on the internal draft version of the Web Terrain Server (WTS) [6] and Web Perspective View Service (WPVS) [7].

Due to the similarity of concepts, this document is partly derived from the Web Map Service Implementation Specification Version 1.3.0 [OGC 06-042] and shares many elements in the GetCapabilities response. The Web Map Service (WMS) GetMap operation influenced the WVS GetView operation.

This document has been aligned to the draft Web 3D Service Implementation (W3DS) Standard Version 0.4.0 [9] and shares many elements in the GetCapabilities response, GetFeature request and response, and GetLayerInfo request and response. The WVS GetView request shares elements with the W3DS GetScene request. Where appropriate, parts from the W3DS specification have been adapted for the WVS specification.

This document includes 6 annexes; Annexes A and B are normative, and Annexes C through F are informative.

Introduction

The Web View Service (WVS) is a portrayal service for three-dimensional geodata such as landscape models, city models, vegetation models, or transportation infrastructure models. A WVS server mainly provides 2D image representing a 3D view on a scene constructed from 3D geodata that is integrated and visualized by the WVS server.

For 2D geovisualization, the Web Map Service (WMS) specifies an accepted and widely used standard that generates and transfers map-like images to a consumer. However, 3D geovisualization faces various challenges: It needs to handle the increased complexity of data models and data size (geometry and texture data), requires high computation resources and graphics capabilities, and requires more complex interaction techniques and hints.

The WVS represents a server-side approach for the visualization, analysis, navigation, and information retrieval for such 3D scenes. This allows for setting up a dedicated server with appropriate 3D graphics hardware and software. On the client-side, users get access to potentially complex 3D geodata with high-quality graphics output and without having to provide and maintain specific 3D graphics hardware and software or streaming complex 3D data – only standard images are transferred. The WVS approach supports thin clients such as mobile phones that do not provide powerful 3D graphics capabilities for processing 3D graphics data. Thus, visual representations of complex 3D scenes can be easily integrated into various workflows, applications, or systems, e.g., web-based applications. Images of 3D views can be easily referenced by URL and transferred, e.g., by email. As neither 3D geodata nor 3D graphics data is transmitted, the WVS inherently supports data security.

Beyond rendering color images, the WVS supports additional image layer, which contain, e.g., depth value or object id information for each pixel of the 3D view. This data can be used for building interactive WVS client applications, which allow users, e.g., to navigate in the 3D view and to interact with it.

Beyond image display, the WVS supports analysis, navigation, and information retrieval for portrayed 3DGeoVEs. Since standard images do not provide thematic data about objects visible in the 3D view, this data has to be accessed by additional service operations. The WVS GetFeatureInfo operation is responsible for accessing additional information on objections and functions equivalent to the WMS.

OpenGIS® Web View Service Implementation Standard

1 Scope

This OGC™ document specifies a draft standard for web services delivering images representing 3D views on a 3D scene created from 3D geodata and allowing for information retrieval from and interaction with these images.

This OGC™ document is applicable to servers with the purpose of visualizing large and complex city and landscape models in a high quality in distributed and heterogeneous environments.

2 Compliance

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

3 Normative references

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

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

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

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

OGC 06-121r3, *OpenGIS Web Services Common Specification.*

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

OGC 08-062r4, *OGC Reference Model*, November 2009.

OGC 04-046r3, *OpenGIS Abstract Specification Topic 2, Spatial Referencing by Coordinates*, August 2004.

OGC 03-105r1, *OpengGIS Geography Markup Language (GML) Implementation Specification*, July 2004.

OGC 09-042, *3D-Symbology Encoding Discussion Draft*, Version 0.0.1, October 2009.

OGC 05-078r4, *OpenGIS Styled Layer Descriptor profile of the Web Map Service Implementation Specification*, Version 1.1.0, June 2007.

OGC 05-077r4, *OpenGIS Symbology Encoding Implementation Specification*, July 2006.

OGC 04-094, *OpenGIS Web Feature Service (WFS) Implementation Specification*, Version 1.1.0, May 2005.

OGC 06-042, *OpenGIS Web Map Server (WMS) Implementation Specification*, Version 1.3.0, March 2006.

OGC 0-061, *OGC Web Terrain Server (WTS),* August 2001.

ISO 8601:2004(E), *Data elements and interchange formats – Information interchange – Representation of dates and times*, December 2004.

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

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

4 Terms and definitions

For the purposes of this standard, the definitions specified in Clause 4 of the OWS Common Implementation Specification [OGC 06-121r3] and in OpenGIS® Abstract Specification shall apply. In addition, the following terms and definitions apply.

4.1
3D scene
computer-graphical representation of a 3D geovirtual environment that is constructed from underlying 3D geodata that is assembled and maintained by a 3D portrayal server, which constructs the 3D scene and could render images that are transferred to a service consumer and can be finally perceived by humans

4.2
3D view
digital image-based representation of a 3D scene, which has been generated for a given projection

4.3
coordinate reference system
coordinate system that is related to the real world by a datum [ISO 19111]

4.4
coordinate system
set of mathematical rules for specifying how coordinates are to be assigned to points [ISO 19111]

4.5
image
set of pixels (picture elements) ordered in a two-dimensional raster generated by sampling, i.e., spatial discretization

4.6
geovirtual environment
a software system for the integration, management, manipulation, analysis, and visualization of geographic information

4.7
image layer
dimension of a 3D view that contains not only color data but also geometric and thematic information such as depth information and object id data

4.8
layer
data layer
dataset
basic unit of geographic information that may be requested from a server

4.9
map
portrayal of geographic information as a digital image file suitable for display on a computer screen

4.10
portrayal
presentation of information to humans [ISO 19117]

4.11
projection
synthetic camera model, which describes how to transform 3D scene data for representation as a two-dimensional 3D view

4.12
scene
collection of graphical elements including geometries, materials, textures, behaviors, animations, lights

4.13
style
visual appearance of a layer in a view

5 Conventions

5.1 Abbreviated terms

Most of the abbreviated terms listed in Subclause 5.1 of the OWS Common Implementation Specification [OGC 06-121r3] apply to this document, plus the following abbreviated terms.

DCE	Distributed Computing Environment
POC	Point of Camera
POI	Point of Interest
SDI	Spatial Data Infrastructure
2D	Two-Dimensional
3D	Three-Dimensional

5.2 UML notation

Most diagrams that appear in this standard are presented using the Unified Modeling Language (UML) static structure diagram, as described in Subclause 5.2 of [OGC 06-121r3].

5.3 Used parts of other documents

This document uses significant parts of document [OGC 06-121r3]. To reduce the need to refer to that document, this document copies some of those parts with small modifications. To indicate those parts to readers of this document, the largely copied parts are shown with a light grey background (15%).

TBR

5.4 Platform-neutral and platform-specific standards

As specified in Clause 10 of OGC Abstract Specification Topic 12 "OpenGIS Service Architecture" (which contains ISO 19119), this document includes both Distributed Computing Platform-neutral and platform-specific standards. This document first specifies each operation request and response in platform-neutral fashion. This is done using a table for each data structure, which lists and defines the parameters and other data structures contained. These tables serve as data dictionaries for the UML model in Annex C, and thus specify the UML model data type and multiplicity of each listed item.

EXAMPLES 1 Platform-neutral standards are contained in Subclauses 9.2.1 and 10.2.1.

The specified platform-neutral data could be encoded in many alternative ways, each appropriate to one or more specific DCPs. This document now specifies encoding appropriate for use of HTTP GET transfer of operations requests (using KVP encoding), and for use of HTTP POST transfer of operations requests (using XML or KVP encoding). However, the same operation requests and responses (and other data) could be encoded for other specific computing platforms, including SOAP/WSDL.

EXAMPLES 2 Platform-specific standards for KVP encoding are contained in Subclauses 9.2.1.6 and 10.2.2.

EXAMPLES 3 Platform-specific standards for XML encoding are contained in Subclauses 9.2.2.2 and 10.2.3.

5.5 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 Table 1.

Table 1 — Contents of data dictionary tables

Column title	Column contents
Names (left column)	Two names for each included parameter or association (or data structure). The first name is the UML model attribute or association role name. The second name uses the XML encoding capitalization specified in Subclause 11.6.2 of [OGC 06-121r3]. The name capitalization rules used are specified in Subclause 11.6.2 of [OGC 06-121r3]. Some names in the tables may appear to contain spaces, but no names contain spaces.
Definition (second column)	Specifies the definition of this parameter (omitting un-necessary words such as “a”, “the”, and “is”). If the parameter value is the identifier of something, not a description or definition, the definition of this parameter should read something like “Identifier of TBD”.
Data type and value (third column) or Data type (if are no second items are included in rows of table)	Normally contains two items: The mandatory first item is often the data type used for this parameter, using data types appropriate in a UML model, in which this parameter is a named attribute of a UML class. Alternately, the first item can identify the data structure (or class) referenced by this association, and references a separate table used to specify the contents of that class (or data structure). The optional second item in the third column of each table should indicate the source of values for this parameter, the alternative values, or other value information, unless the values are quite clear from other listed information.
Multiplicity and use (right or fourth column) or Multiplicity (if are no second items are included in rows of table)	Normally contains two items: The mandatory first item specifies the multiplicity and optionality of this parameter in this data structure, either “One (mandatory)”, “One or more (mandatory)”, “Zero or one (optional)”, or “Zero or more (optional)”. The second item in the right column of each table should specify how any multiplicity other than “One (mandatory)” shall be used. If that parameter is optional, under what condition(s) shall that parameter be included or not included? If that parameter can be repeated, for what is that parameter repeated?

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

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

The contents of these data dictionary tables are normative, including any table footnotes.

6 WVS overview

The specified WVS generates and provides 3D views on 3D geodata such as 3D landscape and city models. It is designed as a 3D portrayal service [ISO 19117] that encapsulates the complete 3D geovisualization pipeline: 3D views are mainly delivered as images, encoded in standard image formats. The WVS encapsulates the whole rendering pipeline and provides mainly images to a service consumer, which could be a client application or another service processing the WVS output within an orchestrated service chain.

The key advantages of this image-based approach include low hardware and software requirements on the service consumer side due to service-side model management, 3D rendering, and moderate bandwidth requirements; only images need to be transferred regardless of the model and rendering complexity.

The WVS is suitable for a thick server - thin client concept, which means that the server collects the necessary geodata, and generates images showing views on this data. A client consuming the WVS is only responsible for displaying the images and, e.g., requesting new images according to user navigation and interaction.

In addition to generating and providing color images, which can be perceived directly by humans, the WVS supports the interactive visualization, exploration, and analysis of the portrayed 3D virtual environment. For this, it provides additional geometrical and thematic information as additional image layers and offers additional operations for retrieving feature information, performing spatial analysis, and supporting navigation.

6.1 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 and proposed [6]. OGC-internally, the discussion led to the Web Perspective View Service (WPVS) [7]. Generalizing the WPVS approach, the Web View Service (WVS) has been specified. Based on a use case analysis [8], the main intension of this generalization and extension is to support a consumer of the image-based 3D portrayal service by means for interacting with the image: navigating in the represented scene, retrieving feature information, and analyzing the spatial setting.

6.2 3D portrayal

6.2.1 Use cases and requirements for 3D portrayal

Portrayal of 3D geospatial data is required for a multitude of systems and applications in various application areas. Annex E lists scenarios for 3D portrayal in the areas of civil services, business development, real estate business, security and safety, tourism, and car and pedestrian navigation [8].

EXAMPLE 1 CIVIL SERVICES — Use a 3D view for processing a building application: A civil service agent can generate a perspective view for a specific address for getting an idea of the current building situation around that address. By picking buildings in the image further building information are displayed.

EXAMPLE 2 TOURISM — Investigate a vacation spot: A tourist can use web-based 3D portrayal for investigating a vacation spot already from home. For example, this could support the decision for a specific accommodation according to the appearance of the surrounding.

Based on such scenarios, the following general requirements can be derived that should be considered by 3D portrayal services for supporting such systems and applications:

a) *Visualization*: As the principal functionality, the visual representation of 3D geospatial data shall be supported. This visualization also serves as the interface to the portrayed data based on various basic interaction techniques. Compared to 2D portrayal, 3D geovisualization is more complex: it requires sophisticated methods and techniques for managing, accessing, and rendering large amounts of geometry and texture data. Specialized rendering hardware and software are required for the generation of appealing visual representations.

b) *Navigation*: Navigation represents the major interaction technique for 3D geovirtual environments. The manipulation of the virtual camera is required for exploring a 3D scene or searching for specific features. Navigation in 3D environments represents a major challenge due to increased degrees of freedom, occlusion problems, and complicated orientation. Here, navigation techniques are required that support users, e.g., by providing and maintaining context information or by interpreting higher-level navigation commands.

c) *Spatial analysis*: Spatial analysis represents another major interaction type in 3D GeoVEs. It allows for understanding, evaluating, and assessing spatial configurations and dimensions. Basic spatial analysis functionalities include distance and area measurement. Specific applications could require additional analysis functionalities.

d) *Information retrieval*: The possibility to retrieving information about portrayed features allows a service consumer to use the visualization as an interface to the “contained” data. It represents a direct and efficient access to the underlying complex 3D data.

e) *Styling*: Styling allows a service consumer to manipulate the appearance of geofeatures in the generated visual representation of a 3D scene. This includes, e.g., the coloring of the surfaces of building representation. Further on, styling depending on feature’s attribute values allows for visual analytics. Beyond feature styling, more general styling mechanism are required for 3D geovirtual environments that specify rendering styles (e.g., illustrative rendering) or the environmental effects (e.g., fog).

The WVS is designed to complement the fundamental capability of image generation and image display by these extended capabilities.

NOTE Styling for 3D is an open question, which is currently targeted by the 3D Symbology Encoding initiative [OGC 09-042].

6.2.2 3D geovisualization pipeline

NOTE The following three paragraphs and Figure 2 are mainly copied from the W3DS discussion paper [OGC 09-104].

Regarding the portrayal of spatial data, the OGC specifies a four-stage visualization pipeline [OGC 08-062r4], which describes visualization as a multi-stage process starting from non-graphical object representations in a repository (e.g., a database) and ending with the presentation of graphical entities on a display device (Figure 1). In the first stage, the selection stage, interesting objects are chosen and retrieved from an underlying data repository. In the second stage, the mapping stage, graphical representations are generated for these objects. In the third step, the generated display elements are rendered to an image, which in the final step is displayed to the user by an appropriate output device. The interaction capabilities described in Section 6.2 represent the possibilities a service consumer has to manipulate the stages of this geovisualization pipeline.

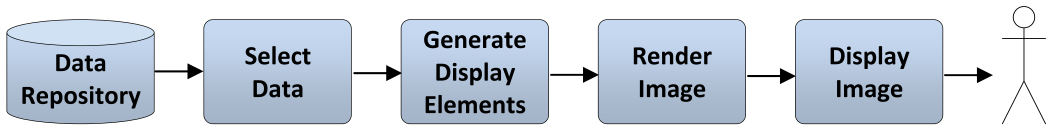


Figure 1 — Stages of the 3D geovisualization pipeline

The components of the portrayal pipeline don’t have to reside on the same system; they can be distributed over the internet. However, in client-server applications the lower level components are typically installed on one or more servers while the remaining visualization tasks are handled by the client. According to their complexity clients are classified into thick, medium, and thin clients (Figure 2).

a) Thick clients communicate on the feature level with the server. The advantage is that the client is free to realize any – including very complex - interaction schemes and analytic functions. The drawback, at least regarding web applications, is the high computing, memory, and bandwidth requirements for processing GML 3D data.

b) Medium clients download display elements which can be easily processed by the graphics hardware. Geometries usually consist of triangle arrays, triangle strips, or indexed triangles. 3D browser plugins are available for X3D. KML can be used by virtual globes.

c) Thin clients only have to cope with rendered images. Hence they are very easy to implement and require no special hardware. In this scheme, a perspective view service could be used for rendering images based on the camera position and viewing direction as specified in the request. Interaction on the client side is reduced but on the other side, sophisticated rendering techniques including ray tracing, ray casting, caustics etc. can be used.

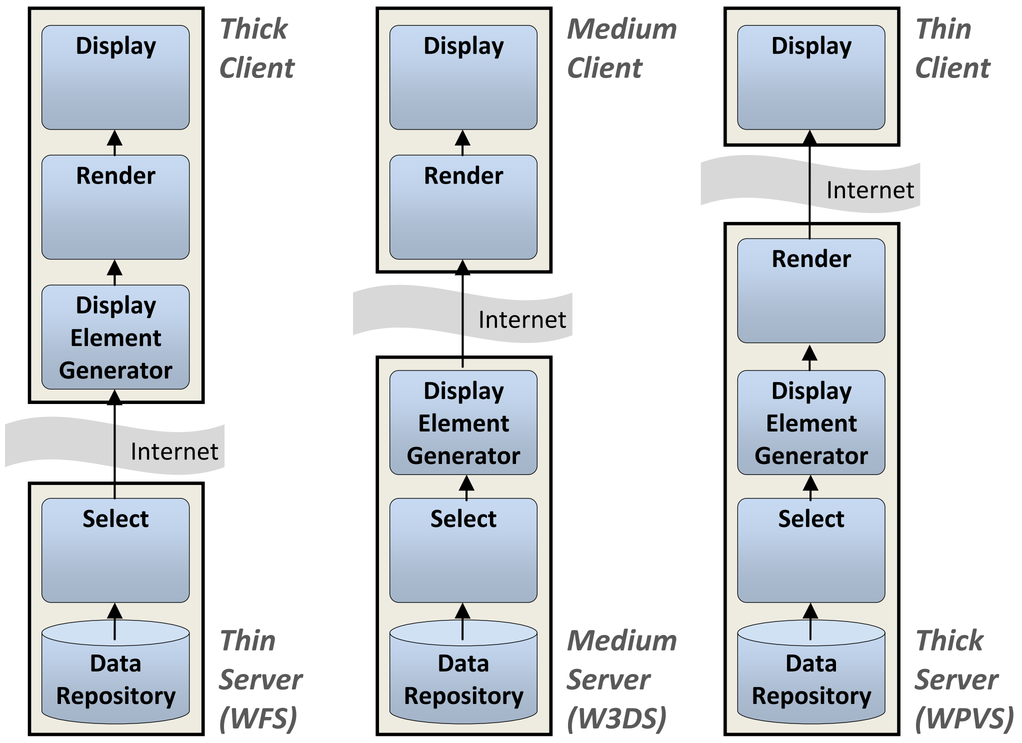


Figure 2 — Balancing schemes between 3D portrayal service servers and clients

The decision which client model is appropriate for what application depends on the specific application scenario on the one hand and the availability of appropriate data and services on the other hand.

The WVS represents a fully server-side approach for the visualization, analysis, navigation, and information retrieval for such 3D scenes. As a key characteristic, only images are transferred between a WVS server and client. This allows for setting up a dedicated rendering server with appropriate 3D graphics hardware and software. On the client-side, users get access to potentially complex 3D geodata with high-quality graphics output and without having to provide and maintain specific 3D graphics hardware and software or streaming complex 3D data – only standard images are transferred.

This approach supports thin clients such as mobile phones that do not provide powerful 3D graphics capabilities for processing 3D graphics data. Thus, visual representations of complex 3D scenes can be easily integrated into various workflows, applications, or systems, e.g., web-based applications. Images of 3D views can be easily referenced by URL and transferred, e.g., by email. By requesting 3D positions at a specific image pixel, a thin WVS client can implement an efficient and targeted step-by-step 3D navigation. Additionally, a thin client can use service-side analysis functionalities (e.g., for measuring distances) or retrieve feature information.

The WVS provides non-color data for a 3D view that can be processed by a more complex client and can be facilitated for reaching real-time 3D navigation, high degrees of interactivity, and potentials for sophisticated post-rendering styling.

In summary, based on transferring images, the WVS supports a range of (interactive) clients. For example, tile-based WVS clients can be implemented, which are based on non-perspective views. Furthermore, the WVS can serve 3D views as a basis for sophisticated 3D spatial analysis, e.g., minimum distance measurements of visual contact analysis. Annex F shows application examples of two different WVS clients.

NOTE Current developments in the field of web-based 3D graphic systems (e.g., Google's O3D, Khronos' WebGL) support browser-based interactive clients accessing a WVS server.

6.3 Definition of a 3D view

In this specification a 3D view describes a digital representation of a 3D scene, which has been created by a specific projection and spatial discretization leading to one or multiple images. By this, the 3D scene is transformed from world CS into image CS. For a discrete pixel of that image-based representation, values of different domains can be computed and stored (Figure 3). For a WVS, color data represents the most prominent information domain of a 3D view. In this specification, images containing values of a specific domain are referred to as image layers. On example is the "COLOR" layer, which contains a color value at each pixel and can be encoded by standard image formats.

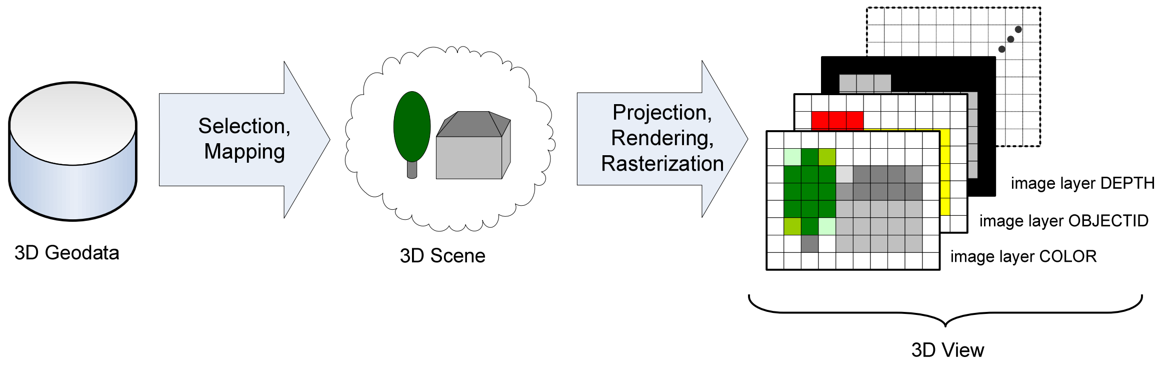


Figure 3 — By projection, rendering, and rasterization, a WVS generates a 3D view containing (multiple) image layers, e.g., COLOR, OBJECTID, or DEPTH

Besides color values, additional geometrical or thematic information can be computed, encoded, and transferred for each 3D view. Examples for these additional image layers are depth data or object id information. A client requesting the WVS can exploit this data for providing an interactive geovisualization to users. E.g., an object id layer can be used for selecting single features in a 3D view.

A 3D view is specified by

d) projection type and parameters, which describe how to project the 3D world to a 2D projection surface (e.g., perspective projections and parallel projections),

e) data layers, i.e., geographic features and phenomena, that shall be portrayed,

f) styling information that shall be applied to the selected data or the overall 3D environment for reaching a specific visual representation (in the case of requesting a color layer),

g) basic portrayal information including dimensions of the 3D view and formats in which to respond the desired view information (e.g., standard images for color information).

6.4 Relationships to other services and data standards

6.4.1 WPVS

WPVS [7] (as specified OGC-internally) and WVS are of the same class of 3D portrayal services. Both services follow the approach of full service-side rendering and providing only images to service consumer.

The WPVS lacks relevant capabilities for the effective usage of the portrayed geodata. The WPVS does not allow for efficiently navigating in the presented 3D scene, for retrieving feature information, or for performing spatial analysis. For example, in a WPVS client, navigation is step-by-step only and far from real-time interaction. A user can not specify a new view based on a previously retrieved view, e.g., positioning the virtual camera at a specific location by clicking into the image.

The WVS is designed to overcome these limitations of interactivity by providing additional geometrical and thematic data for a 3D view, and by providing additional operations. A WVS allows for implementing interactive thin clients that allow for efficiently use the 3D geoinformation presented by the service.

6.4.2 Other services and data formats

A WVS can be integrated with other OGC services providing geodata or graphics data (Figure 4). Besides hosting and using internal geodata, a WVS could access remote data from W3DS, WMS, or WFS. Accessing remote data could also be triggered by user-defined styling within a WVS GetView request (SLD-enabled WVS only). Additionally, a GetFeatureInfo response could provide feature identifiers for allowing a WVS consumer to request complete feature data from a remote WFS.

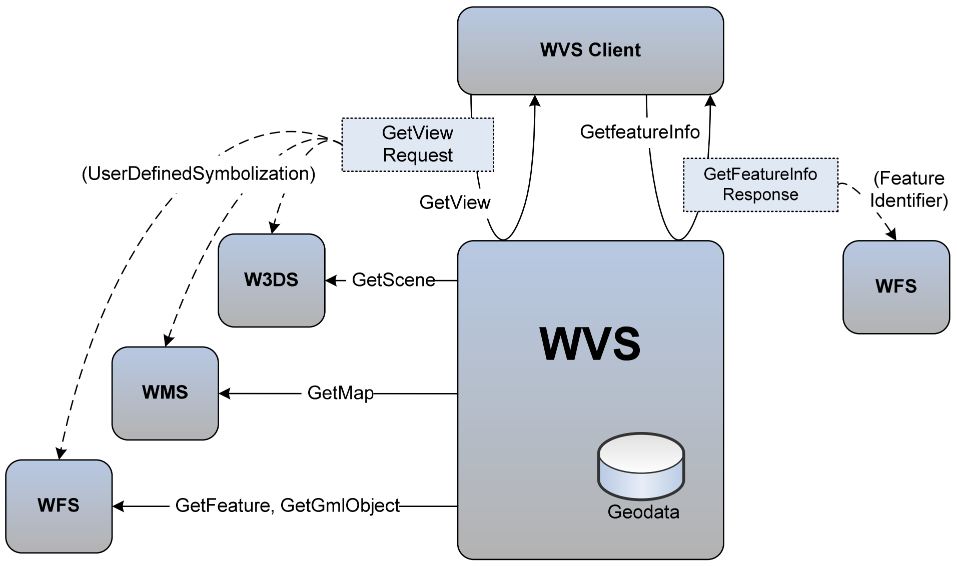


Figure 4 — Possible integration of the WVS with other OGC web services

6.4.2.1 W3DS

The W3DS mainly generates and delivers 3D computer graphical representations, which typically are arranged as scene graphs (including, e.g., geometries and textures) and are transmitted to the service consumer using 3D graphics formats (e.g., VRML, X3D, COLLADA). This graphics data can include appearance information but is mostly not capable of carrying thematic information. Medium-size W3DS clients have to process, manage, and render this data (Figure 1). A W3DS client is a medium client, which generally supports real-time navigation and interaction in the 3D scene.

Table 2 — Comparison of W3DS and WVS

	W3DS	WVS
Transferred data	General scene graphs	Images
Transmission load	Depends on the size of the 3D model and required quality, e.g., when including textures	Depends on the type and size of the image
Separation of rendering concerns	Medium servers – medium clients	Thick servers – thin clients
Resulting visual quality	Depends on the client’s computing and rendering capabilities	Only depends on the servers capabilities
Client’s interactivity	Real-time navigation possible	Real-time navigation possible by medium clients

Table 1 briefly compares the W3DS and WVS. From these characteristics different usages of both services follow. Compared to the W3DS, the WVS also supports light-weight clients that do not possess enough computing and 3D graphics capabilities for consuming, managing, and rendering massive graphics data. For retrieving complex scenes from a W3DS a high bandwidth is needed; the WVS allows for high-quality images of even in situations where available bandwidth is low.

WVS and W3DS share the concept of arranging feature data in layers. Thus, in a spatial data infrastructure (SDI) that provides 3D portrayal, a WVS server could act as a consumer of a W3DS server. For example, the WVS could internally use graphics data prepared and delivered by a W3DS. Additionally, in the context of user-defined styling, an SLD-enabled WVS could accept a request to a (remote) W3DS as part of the styling description of a WVS GetView request, load this data, and integrate it in the generated 3D view.

6.4.2.2 Geodata formats and graphics data formats

Open geodata standards such as CityGML provide formats for the interoperable exchange of geodata between different services, systems, and applications in a spatial data infrastructure. A WVS could support such open standards as output format for the GetFeatureInfo operation that allows a service consumer to retrieve feature information for the geo objects at a specific image pixel.

Combined with appropriate styling mechanisms, geodata formats such as CityGML or graphics data formats such as KML, Collada, or X3D could serve as input for a an SLD-enabled WVS. Such SLD-enabled WVS could integrate the geodata or graphics data provided with the styling information of a GetView request with the WVS data repository for allowing for user-defined styling.

6.4.2.3 WFS

A WFS provides access to raw geodata [OGC 04-094], which is delivered as collection of feature data. The data to request can be selected by applying complex filters.

NOTE For keeping separations of concerns, a WVS is not intended to provide the functionality of WFS, i.e., to provide the underlying geodata. If this functionality is required, a separate WFS server should be installed.

As part of the feature information retrieved by the WFS GetFeatureInfo operation, a WVS could provide the feature identifier, which could be used by consumer to retrieve the original geodata from a WFS server by using its GetGmlObject or GetFeature operations.

Internally, a WVS server could use a WFS server as data source. For example, a WFS could serve geofeatures (e.g., building data) that would be processed by the WVS server. Additionally, styling information within a GetView request (SLD-enabled WVS only) could include A) requests to (remote) WFS server from where to fetch geodata and B) information of how to graphically represent this data, which then could be used by the WVS to integrate this data.

6.4.2.4 WMS

The WMS [OGC 06-042] is the widely accepted and used standard for image-based 2D portrayal. The WMS provides an easy to use interface for retrieving map-like representations of complex geodata. A WVS serves as a counterpart for portraying complex 3D geodata.

Internally, a WVS server could use a WMS as data source. For example, a WMS could serve map-like representations that can be used as terrain textures. Additionally, styling information within a GetView request (SLD-enabled WVS only) could include WMS requests, e.g., for specifying the appearance of a terrain model layer.

6.5 Data layer concept

The WVS generates 3D views that include data from a specified set of data “layers”.

NOTE Next paragraph is from draft W3DS specification [9].

This layer concept is not exactly the same as used in the WMS. In WMS a layer can be considered as a transparent sheet with features drawn upon it. The order in which layers are drawn on the map is important because the features may overlap. The order determines what will be visible and what will be hidden. In 3D scenes, the order in which features are drawn on the screen is irrelevant and the visibility is determined solely by the distance to the virtual viewpoint. Features overlap based on the spatial position of the triangles which make up the feature geometries. The layer concept is comparable to the FeatureType in WFS. However, we keep the term layer because it is commonly understandable and in order to make a distinction between WFS, which serves feature for being processed in GIS software and portrayal services, which serve images and display elements for end users.

This means, a WVS layer contains features to portray according to a specific style. A layer contains features of one feature type. For example, a layer “buildings” would contain only CityGML building features. – In that concept, maps and coverages are integrated as appearances of features, e.g., as a terrain texture of a “terrain” data layer.

QUESTION Should the WVS data layer also include map and coverage data, which, .e.g., could are draped on specific (classified) layers such as a terrain layer?

6.6 Image layer concept

Besides color layers, the WVS supports additional image layers of 3D view, which store various spatial and thematic information for each image pixel [3]. This data does not necessarily represent color values and is not necessarily dedicated for human cognition.

However, the WVS supports encoding also non-color image layers by standard image formats. This means to apply for all image layers the same principles for data encoding, data exchange, and client-side data loading and processing. Additionally, using image encodings allows for applying state-of-the-art compression algorithms.

COMMENT Besides encoding view-related data as color values, spatial and thematic image data could be also advertised and transferred in non-image formats, e.g., text/xml.

This WVS specification suggests A) relevant image layer types and B) relevant encodings of this view-related data, e.g., as color values of color images.

6.6.1 Image layer types

Figure 5 shows examples of each image layer type that is suggested by the WVS.

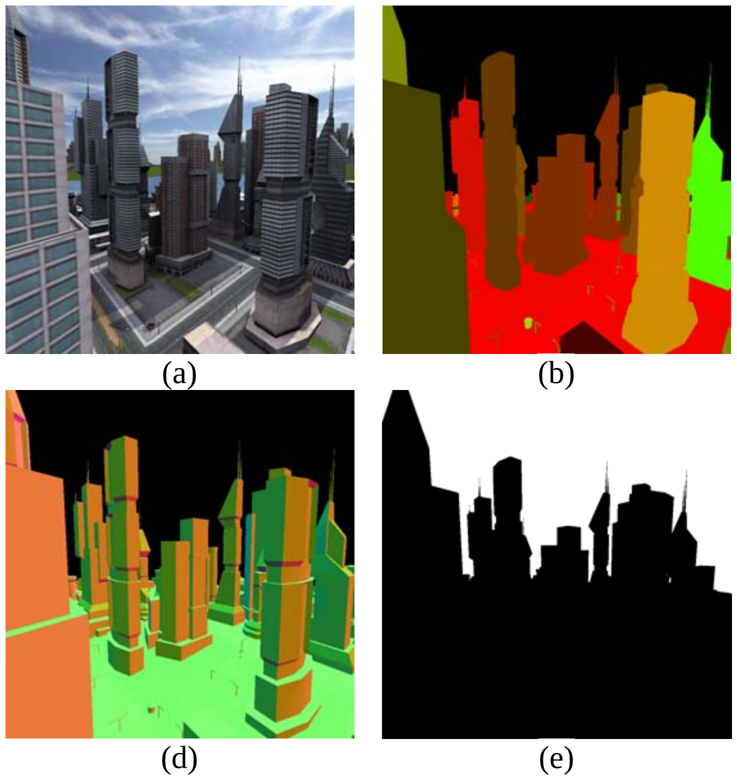


Figure 5 — Image layers, which can be rendered and provided by a WVS: Color layer (a), depth layer (b), object id layer (c), normal layer (d), mask layer (e)

6.6.1.1 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 images encodings provide good compression results.

NOTE For storing transparent values, image formats supporting transparency are required. PNG images are one example of such formats.

COMMENT Maybe color layers should also support parameters, e.g., for adjusting lighting and only retrieving diffuse colors.

6.6.1.2 Depth layer

A depth layer describes the distance to the virtual camera for each pixel. 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.

NOTE 1 This depth layer does not provide the z-value used for rendering but real world distances in meters.

NOTE 2 Depth values may have a different meaning for different projection types. For example, for an orthographic projection, the depth value represents the distance to the projection plane.

Depth images can, e.g., serve for multiple visual effects such as depth-of-field effects. Furthermore, depth images represent a major means to compose multiple images generated from the same camera position and by the same projection.

6.6.1.3 Object id layer

Object ids are distinct numeric values assigned to all the features in the 3DGeoVE provided by a WVS. An object id is assigned to any top-level feature of a WVS data layer. An ObjectId layer contains an object id for each pixel. For a WVS server, object ids shall be unique over all provided data Layers and even over multiple GetView request, due to facilitating consistent interaction across multiple 3D views.

NOTE 1 A data layer “buildings” may contain building features that contain subfeatures (e.g., walls, doors, etc.) An object id layer for this layer assigns object ids only to the building features. Thus, subfeatures can not be distinguished in an object id image.

COMMENT Should the WVS provide a mechanism for identifying subfeatures, too?

Object id value “0” is reserved for pixels that do not represent a feature (e.g., for pixels representing the sky) or for which no object id could be determined.

It is left to a specific WVS server how to determine and assign unique object ids to all features. Approaches for this include computing hash-values based on, e.g., A) the feature type and feature identifier or B) a set of characteristics such as the feature’s data layer, bounding box, and creation data.

A WVS consumer requesting an ObjectId layer has to decode these Byte representations and restore the integer value. Using the object ids provided with the ObjectId layer, a client application can, e.g., determine all the pixels that show a specific feature, e.g., for highlighting, contouring, or spatially analyzing this feature.

An object id only refers to the graphical representation of a feature, it is not meant as a feature identifier. The WVS operation GetIdentifierMapping allows for determining feature identifiers from object identifiers. This feature identifier, e.g., could be used for requesting an underlying WFS (if advertised in the data layers metadata).

NOTE 2 When allowing consumers to provide own data to be visualized, the consistent generation and management of object ids could become non-trivial.

6.6.1.4 Normal layer

A normal layer describes for each pixel the direction of the surface normal visible at that pixel. A normal layer could be used, e.g., for client-side, subsequent integration of additional light sources and the adjustment of color values around pixels, e.g., for highlighting scene objects.

6.6.1.5 Mask layer

A mask layer contains a value of 1 for each pixel that covers a scene object and 0 otherwise. Mask layers could, e.g., provide information about unused image space or could support the altering of the scene background.

6.6.1.6 Other image layers

The image layers specified in this document represent a fundamental subset of view-related data. However, image layers are not limited to the ones specified in clauses 6.6.1.1 - 6.6.1.5. A WVS server can implement and provide other image layers. Additional image layers shall be advertised in the WVS server's metadata document together with available encodings and formats.

6.6.2 Default 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 server's metadata document.

WVS suggests representing image layer data by colors and encoding them by appropriate standard image formats. Alternative encodings/formats are possible.

In clauses 6.6.2.1 - 6.6.2.4, default encodings for the suggested image layers are described. Table 3 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 3 — Examples of MIME types for image data encoding

Image Layer Identifier	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 Layer Identifier	Mime types
	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 server-specific data encodings, a Mime type parameter "encoding" could be used for identifying the data encoding and format.

EXAMPLE WVS server-specific format for a normal layer encoding could be "image/png; mode=24bit, encoding=two-component" which could, e.g., specify the representation of normal vectors by two components only.

6.6.2.1 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 6 shows an example of encoding depth values as 32 bit RGBA colors.

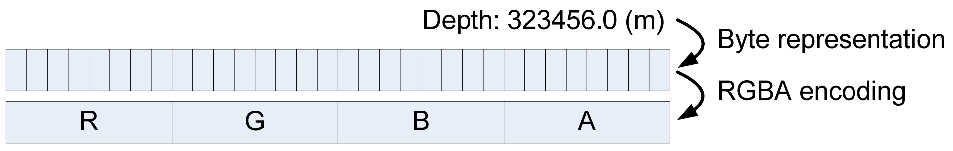


Figure 6 — Encoding a depth value as 32 bit RGBA color components

NOTE Due to the direct storage as color components, resulting depth images posses very little pixel-to-pixel coherence, and should not be stored by lossy image formats.

6.6.2.2 Object id 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 7 shows an example of encoding object ids as 32 bit RGBA colors.

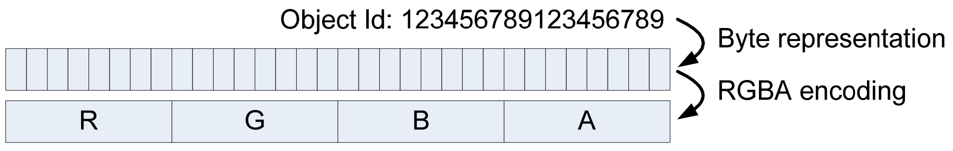


Figure 7 — Encoding an object id as 32 bit RGBA color components

6.6.2.3 Normal layer default encodings

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 WVS 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.

6.6.2.4 Mask layer default encoding

Mask values 0 and 1 shall be encoded by the colors white, representing pixels that do not cover a scene object (0), and black for pixels that cover scene objects (1). Due to this, mask layers can be encoded in any image format. Black and white images serve well, due to reduced image size.

6.7 Image styling

Symbology Encoding (SE) [OGC 05-077r4] provides capabilities for styling, e.g., features and coverages in a 2D map – 3D styling capabilities are not yet considered. Currently, 3D extensions to SE are discussed within the OGC [OGC 09-042].

Until these extensions become an OGC standard, the WVS provides a mechanism for styling the overall finally rendered image based on named ImageStyles, in contrast to feature specific styling provided by the GetView Styles parameter.

ImageStyles can encapsulate and provide scene-wide styling or complex rendering techniques, such as environmental effects like fog and illustrative, non-photo realistic rendering, respectively.

6.8 3D projections

Two-dimensional representations of a 3D virtual environment are generated by transforming points of the 3D scene onto a projection surface. Such projections can be categorized into perspective projections and parallel projections [2], which can be further subcategorized, e.g., in one and two-point perspective projections or orthographic and oblique parallel projections (Figure 8).

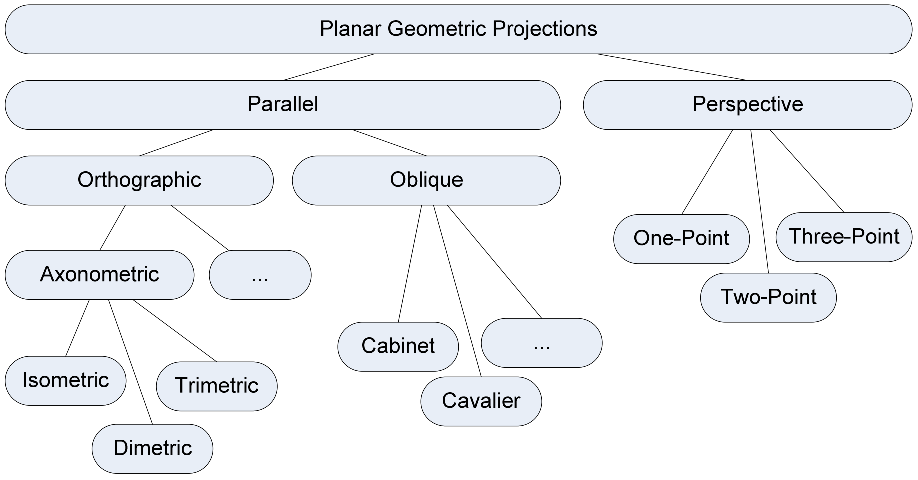


Figure 8 — Classification of planar geometric projections

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 centre of projection are collinear. With non-planar projections, these rays from centre of projection to the three-dimensional point are non-linearly mapped, but are, e.g., projected spherical or cylindrical.

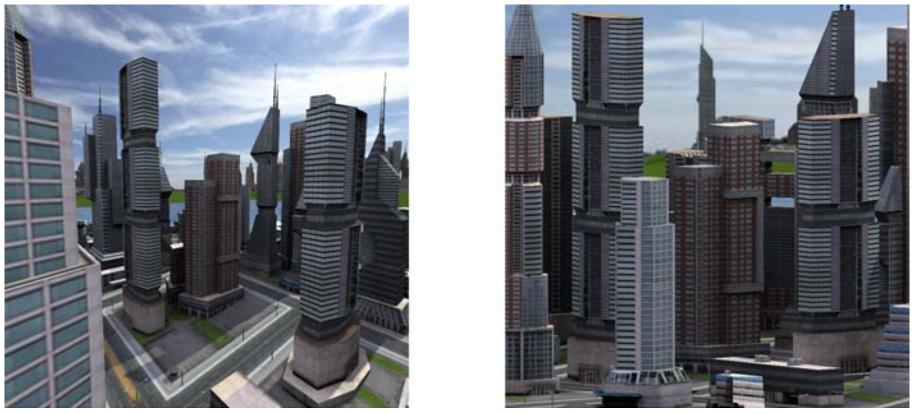


Figure 9 — Examples of perspective (left) and orthographic projections (right).

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 WVS allows for choosing a projection type that shall be applied for rendering the view; currently the WVS specifies two projection types, perspective projection and orthographic projection (Figure 9).

NOTE Additional projection models could be described by a specific WVS.

6.9 Coordinate Systems

6.9.1 Introduction

This standard uses two principal classes of Coordinate Systems: an Image CS applicable to the image showing the 3D view generated by the WVS, and a Layer CRS for a Bounding Box applied to the source data.

6.9.2 Image CS

An Image CS is a coordinate system for a 3D view produced by a WVS. A WVS 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 the GetView request (9.2.1), GetFeatureInfo request (10.2.1), GetPosition request (12.2.1), GetMeasurement request (13.2.1), and GetCamera request (14.2.1) 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*).

6.9.3 Layer CRS

TBD — see WMS, version 1.3.0, Section 6.7.3 [OGC 06-042]

6.10 Extensibility

The WVS is designed for supporting extensibility of the following aspects of 3D portrayal:

a) Projections: Beyond perspective and orthographic projections, specific WVS servers can provide additional projection types. Those shall be advertised in the WVS server’s metadata document.

EXAMPLE 1 A specific WVS server could provide isometric projections.

b) Image layers: Beyond image layers COLOR, OBJECTID, DEPTH, NORMAL, MASK, specific WVS servers can provide additional image layers. Those shall be advertised in the WVS server’s metadata document.

EXAMPLE 2 A specific WVS server could provide an image layer containing only the specular lighting information, which could be used, e.g., for image post-processing.

c) Image layer encodings: Beyond encoding image layers as suggested in this specification, specific WVS server can provide other encodings. These encodings shall be advertised in the WVS server's metadata document.

6.11 WVS interface

The WVS interface (currently) specifies 9 operations that can be requested by a client and performed by a WVS server. Those operations are:

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

b) GetView (required implementation by servers) – This operation allows a client to retrieve a 3D view on a 3D scene.

c) GetFeatureInfo (optional implementation by servers) – This operation allows a client to retrieve attribute data of features selected in a 3D view.

d) GetIdentifierMapping (optional implementation by servers) – This operation allows a client to retrieve feature identifiers, e.g., for the features visible in a 3D view.

e) GetPosition (optional implementation by servers) – This operation allows a client to retrieve the 3D positions at a specified 2D pixel position of a requested view or, vice versa, a 2D pixel position representing a 3D position in the scene.

f) GetMeasurement (optional implementation by servers) – This operation allows a client to perform measurement and retrieve the measurement result.

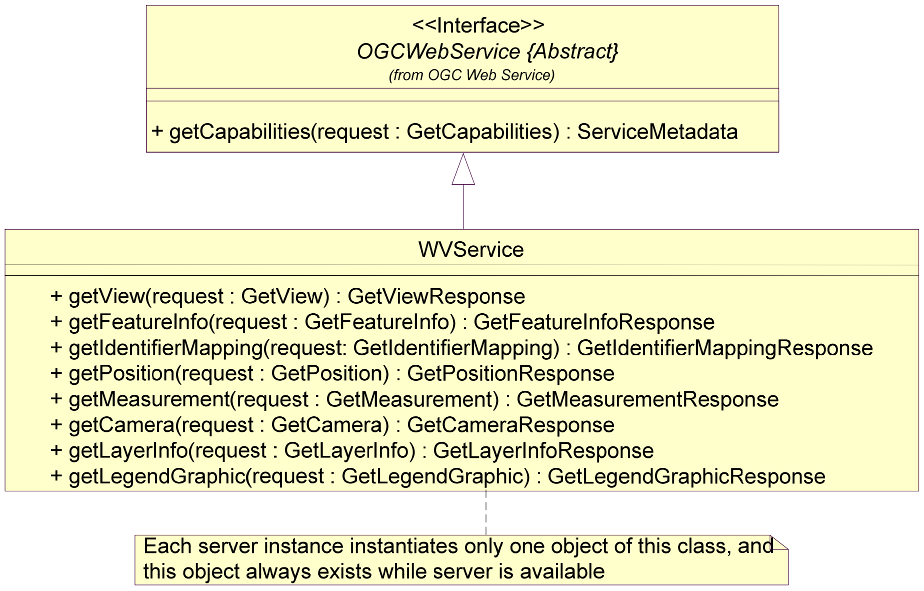
g) GetCamera (optional implementation by servers) – This operation allows a client to retrieve a "good" camera specification based on the current image, i.e. this operation allows for server-side navigation support.

h) GetLayerInfo (optional implementation by servers) – This operation allows a client to retrieve information on available attribute names and values of a selected layer.

i) GetLegendGraphic (optional implementation by servers) – This operation allows a client to retrieve a legend graphic for a specific feature and styling.

These operations have many similarities to other OGC Web Services, including the WMS, WFS, and WCS. Many of these interface aspects that are common with other OWSs are thus specified in the OpenGIS® Web Services Common Implementation Specification [OGC 06-121r3]. Many of these common aspects are normatively referenced herein, instead of being repeated in this standard.

Figure 10 is a simple UML diagram summarizing the WVS interface. This class diagram shows that the WVS interface class inherits the getCapabilities operation from the OGCWebService interface class, and adds the getView, getFeatureInfo, getIdentifierMapping, getPosition, getMeasurement, getCamera, getLayerInfo, and getLegendGraphic operations. (This capitalization of names uses the OGC/ISO profile of UML.) A more complete UML model of the WVS interface is provided in Annex C (informative).



<<Interface>>
OGCWebService {Abstract}
(from OGC Web Service)
+ getCapabilities(request : GetCapabilities) : ServiceMetadata
WVService
+ getView(request : GetView) : GetViewResponse
+ getFeatureInfo(request : GetFeatureInfo) : GetFeatureInfoResponse
+ getIdentifierMapping(request: GetIdentifierMapping) : GetIdentifierMappingResponse
+ getPosition(request : GetPosition) : GetPositionResponse
+ getMeasurement(request : GetMeasurement) : GetMeasurementResponse
+ getCamera(request : GetCamera) : GetCameraResponse
+ getLayerInfo(request : GetLayerInfo) : GetLayerInfoResponse
+ getLegendGraphic(request : GetLegendGraphic) : GetLegendGraphicResponse
Each server instance instantiates only one object of this class, and this object always exists while server is available

Figure 10 — WVS interface UML diagram

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

Each of the WVS operations is described in more detail in subsequent clauses.

7 Shared aspects

7.1 Introduction

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

7.2 Shared operation parameters

This clause specifies some of the parameters used by multiple operations specified in the following clauses. The parameter names, meanings, data types, and multiplicity shall be as specified in Table 4.

Table 4 — Definitions of some operation request and response parameters

Names	Definition	Data type and value	Multiplicity and use
crs CRS	CRS to apply to BoundingBox, Projection parameters, and 2D and 3D Positions [a]	URI [b]	One (mandatory)
dataSelection DataSelection	Data selection and subset selection	DataSelection type See Table 5	One (mandatory)
dataStyling DataStyling	Layer-specific styling, feature-specific styling, and overall view styling	DataStyling type See Table 7	One (mandatory)
projection Projection	Camera and projection specification for generating images and retrieving scene-related data	ProjectionBaseType E.g., PerspectiveProjection or OrthographicProjection, see Table 8 and Table 9	Zero or more Not each operation request includes Projection data structure, others shall contain exactly one or more
exceptions Exceptions	Reference to format in which operation exceptions should be returned [c]	CodeList type, either: “XML”, “INIMAGE”, “BLANK” Default is “XML”	Zero or one (optional) Include when exception format other than “XML” desired

a In general, this CRS may be 2D horizontal, 3D horizontal plus vertical, or 4D horizontal plus vertical plus time. A 4D CRS shall be a compound CRS, and a 3D CRS may be a compound CRS.

b The CRS shall be specified using either the European Survey Group form “EPSG:<POSC Code>” or the URL format defined OWS Common. Examples: “EPSG:31467”, “urn:ogc:def:crs:EPSG:6.3:31467”, see [OGC 06-121r3] clause 10.3 and [OGC 04-046r3].

c The Exceptions parameter is only available for the GetView and GetLegendGraphic operations. Other operations shall always respond errors in XML format.

7.2.1 CRS

The parameter value for the coordinate reference system (CRS) is defined in [OGC 06-121r3] clause 10.3 and [OGC 04-046r3].

7.2.2 Data selection

Data selection refers to selecting those features that shall be included in the generated 3D view. Data selection is mainly done by choosing layers which to include. Additionally, a WVS consumer could select a spatial subset of this data by specifying a bounding box.

The parameter names, meanings, data types, and multiplicity shall be as specified in Table 5.

Table 5 — Parameters in DataSelection data structure

Names [a]	Definition	Data type and values	Multiplicity and use
boundingBox BoundingBox	Bounding box surrounding desired subset of layer(s), in desired CRS	BoundingBox data structure, see [OGC 06-121r3] clause 10.2 [f]	Zero or one (optional) Include when spatial selection desired
spatialSelection SpatialSelection	Indicates method of selecting objects with BoundingBox	Character String type, not empty Value is one of “overlaps”, “contains_center”, “cut”.	Zero or one (optional) Include when spatial selection desired
time Time	Date and time	Date type, not empty. Data type defined in [ISO 8601:2004(E)] Default is “current”	Zero or one (optional) Include when temporal selection desired
layers Layers	List of Identifier(s) of desired data layer(s)	List of Character String type, not empty Values are specified in service metadata	One (optional)

7.2.2.1 BoundingBox and SpatialSelection

The optional BoundingBox parameter allows a Client to include only a particular spatial subset of geodata for being included in the 3D view, which requires that the WVS server support the spatial selection of features.

The optional SpatialSelection parameter specifies the method for selecting features to be rendered and included in the 3D view based on the (optional) BoundingBox parameter. Table 6 lists and describes the supported spatial selection methods.

Table 6 — SpatialSelection values and meaning

Method name	Definition
overlaps	Checking the spatial relation between the BoundingBox and the feature’s extent. If the feature geometry is contained in or intersects with the BoundingBox, then the feature is selected.
contains_center	Checking whether the feature’s center point is contained or intersects with the BoundingBox. How the center point is computed by the server is not defined, but it shall be inside the convex hull of the feature.
cut	This spatial selection method shall not select features or part of features that lie outside of the BoundingBox. Features that are completely contained in the BoundingBox shall be selected unmodified. Features that intersect with the borders of the BoundingBox shall be cut at this border: Parts that lie outside shall be culled.

A WVS server shall advertise the supported spatial selection methods by the <SupportedSpatialSelections> element in the server’s metadata document and shall omit the <SupportedSpatialSelections> element if it does not support spatial selections.

If a WVS server does not support and advertise spatial selection based on the bounding box but receives a request containing BoundingBox or SpatialSelection parameter, it shall raise a SpatialSelectionNotSupported exception (“locator=BoundingBox” or “locator=SpatialSelection” respectively).

If a WVS server receives a request containing a SpatialSelection parameter value that is not supported and advertised, it shall raise an InvalidParameterValue exception (locator=”SpatialSelection”).

If a request contains an invalid BoundingBox (e.g., one whose minimum X is greater than or equal to the maximum X, or whose minimum Y is greater than or equal to the maximum Y) the server shall throw an InvalidParameterValue exception (locator = “BoundingBox”). If the BoundingBox values are not defined for the given CRS (e.g., latitudes greater than 90 degrees in CRS:84), the server should only select data that is inside the valid range of the CRS for generating the 3D view.

If a WVS server receives a request containing only one of BoundingBox or SpatialSelection parameters, it shall raise a MissingParameterValue exception having the name of the absent parameter as locator string.

If a WVS server receives a request containing neither BoundingBox nor SpatialSelection parameter, it should include all the features visible in the view frustum, which is bounded by the NearPlane and FarPlane parameters.

7.2.2.2 Time

The Time parameter can be used to include a temporal criterion to the feature selection method. The server shall select only features that exist at the given point in time if the layer metadata contains StartTime and/or EndTime elements. In this case the Time parameter includes a data which may be the present system time of a historical date.

The specification of valid parameter values is in accordance with [ISO 8601:2004(E)]. The special keyword “Time=current” may be used to indicate the most current data available at the current timestamp. This is also the default value.

Examples:

```
Example 1: Time=2003-10-25
Specifies the 25th of October 2003

Example 2: Time=2003-10-25T14:28:43Z
Specifies the 25th of October 2003 at 14:28 hours and 43 seconds. Z
means Zulu time which is equivalent to UTC.

Example 3: Time=-1279
Means year 1279 B.C. Ramses II becomes pharaoh of Egypt
```

7.2.2.3 Layers

The mandatory Layers parameter specifies a list of layer identifiers from which to select features for generating the 3D view. The concept of the layer is a metaphor to the traditional (two-dimensional) cartography, with which features of different classes were drawn on different foils resulting in a map with an overall view of these foils. The order in which the layers are listed in the Layers parameter does not influence the visual appearance of the generated scene. However, the order of the lists in the (optional) Styles parameter shall correlate with the Layers list. Each entry in the Layers list shall refer to a layer identifier as described in the server's metadata.

7.2.3 Data styling

Data styling refers to styling specific features and/or output images. Feature styling could modify the geometric complexity of portrayed features, e.g., by selecting different levels-of-detail (LODs). Thus, data styling information needs to be included not only into operation requests for generating visual representations of a 3D view but also into operation requests for retrieving information about a 3D view, e.g. for retrieving feature data for a specific view. The parameter names, meanings, data types, and multiplicity shall be as specified in Table 7.

Table 7 — Parameters in DataStyling data structure

Names [a]	**Definition**	**Data type and values**	**Multiplicity and use**
styles Styles	List of Identifier(s) of desired layer style(s)	List of CharacterString type, can be empty. List must have same length as in parameter Layers Values are specified in service metadata	Zero or one (optional) Include when named styles desired
imageStyles ImageStyles	Identifier(s) of desired overall image style(s)	List of CharacterString type, not empty Values are specified in service metadata	Zero or one (optional) Include when image styling desired
sld SLD	Styled layer descriptor file	URL or file contents	Zero or one (optional)
sld_body SLD_Body	SLD document in HTTP GET request	<StyledLayerDescriptor> TBD: As in "SLD profile for WMS" [OGC 05-078r4] spec?	Zero or one (optional) Include when SLD document in HTTP GET request desired

COMMENT With user-defined styling, the "Layers" parameter of the DataSelection data structure should become optional.

7.2.3.1 Styles

The optional Styles parameter specifies for each Layer, which style shall be applied by the server when generating display elements for rendering the final 3D view. The Styles parameter contains a list of style identifiers. The length of this list shall be equal to the length of the list in the Layers parameter of the DataSelection element. The order of this list shall correlate with the list in the Layers parameter, meaning that style *n* shall be applied to layer *n*. Each entry in this list shall refer to a style identifier as described in the server’s metadata. Empty list entries may be used to indicate the default style. An empty value for the Styles parameter may be used to indicate default styles for each layer in the Layers list.

COMMENT Should the parameter Styles better be renamed to LayerStyles?

7.2.3.2 ImageStyles

The optional ImageStyles parameter specifies, which styles to apply to the overall 3D view image. The ImageStyles parameter contains a list of ImageStyle identifiers. Each entry in this list shall refer to an ImageStyle identifier as described in the server’s metadata.

7.2.4 Projection

The WVS specifies two planar projection types, perspective projections and orthographic projections. The parameter names, meanings, data types, and multiplicity shall be as specified in Table 8 or Table 9. Possible values for the parameters of each projection type should be advertised in the server’s metadata.

If a WVS server receives a request that contains invalid Projection parameters it shall raise an InvalidProjection exception having the parameter name as locator.

A specific WVS server can provide additional projection types. For this, the projection shall be advertised in the server’s metadata document.

Table 8 — Parameters in PerspectiveProjection data structure

Names [a]	Definition	Data type and values	Multiplicity and use
POC POC	Position of the virtual camera	Position3D data structure, see Table 11	One (mandatory)
POI POI	Position of point of interest	Position3D data structure, see Table 11	One (mandatory)
up Up	Normal vector pointing in up direction	Position3D data structure, see Table 11	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 degree, server 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 degree, server 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 server 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 server 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 9 — Parameters in OrthographicProjection data structure

Names [a]	Definition	Data type and values	Multiplicity and use
POC POC	Position of the virtual camera	Position3D data structure, see Table 11	One (mandatory)
POI POI	Position of point of interest	Position3D data structure, see Table 11	One (mandatory)
up Up	Normal vector pointing in up direction	Position3D data structure, see Table 11	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 server 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 server 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 server metadata. Value in measure of the request CRS [a]	Zero or one (optional) Include when value other than default desired
top Top	Distance to the top border of the view frustum	Double type Default value is advertised in server 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 server metadata. Values in the measure of the request CRS [a]	Zero or one (optional) Include when near plane other than default desired
farPlane FarPlane	Distance to the far clipping plane	Double type Default value is advertised in server metadata. Values in measure of the request CRS [a]	Zero or one (optional) Include when far plane than default desired
a In the case of a 2D CRS, measure unit is meter			

Table 10 — Parameters in Position2D data structure

Names	Definition	Data type and values	Multiplicity and use
x X	First position coordinate, in request CRS	Number type Origin and units specified by CRS	One (mandatory)
y Y	Second position coordinate, in request CRS	Number type Origin and units specified by CRS	One (mandatory)

Table 11 — Parameters in Position3D data structure

Names	Definition	Data type and values	Multiplicity and use
x X	First position coordinate, in request CRS	Number type Origin and units specified by CRS	One (mandatory)
y Y	Second position coordinate, in request CRS	Number type Origin and units specified by CRS	One (mandatory)
z Z	Third position coordinate, in request CRS	Number type Origin and units specified by CRS	One (mandatory)

7.2.4.1 POC, POI, Up

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

7.2.4.2 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 WVS server shall suggest appropriate default values in the <NearPlaneHint> and <FarPlaneHint> elements of the server metadata.

7.2.4.3 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 server shall suggest default values for FOVX and FOVY. If FOVX and FOVY are not provided, the server shall use default values. If only one of FOVX or FOVY is provided, the server shall compute the missing value from the aspect ration of the requested Width and Height.

7.2.4.4 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

7.2.5 Exceptions

The operations GetView and GetLegendGraphic support an optional Exceptions parameter, which states the format in which to report errors during processing a request. The default value is “XML” if this parameter is absent from the request.

A WVS shall offer one or more of the following exception reporting formats by listing them in separate <Format> elements inside the <Exceptions> element of the operations metadata. A WVS server shall implement and offer at least the "XML" exception format; the other values are optional.

NOTE For all other operations, a WVS shall encode exception reports in text/xml format.

7.2.5.1.1 XML (mandatory)

Errors are reported using service exception XML, as specified in [OGC 06-121r3], Clause 8. This is the default exception format if none is specified in the request. The remaining exception formats are optional. A server may issue a service exception in the default XML format if a request specifies a different exception format not supported by the server.

7.2.5.1.2 INIMAGE (optional)

If the Exceptions parameter is set to INIMAGE, the WVS 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.

7.2.5.1.3 BLANK (optional)

If the Exceptions parameter is set to BLANK, the WVS shall, upon detecting an error, return an object of the type specified in FORMAT whose content is uniformly "off". In the case of a picture format, that response shall be an image containing only pixels of one color (the BackgroundColor).

7.3 Operation request encoding

The encoding of operation requests shall use HTTP GET with KVP encoding and HTTP POST with XML encoding as specified in Clause 11 of [OGC 06-121r3]. Table 12 summarizes the WVS Service operations and their encoding methods defined in this standard.

Table 12 — Operation request encoding

Operation name	Request encoding
GetCapabilities (required)	KVP and optional XML
GetView	XML and optional KVP
GetFeatureInfo	XML and optional KVP
GetIdentifierMapping	XML and optional KVP
GetPosition	XML and optional KVP
GetMeasurement	XML and optional KVP
GetCamera	XML and optional KVP
GetLayerInfo	XML and optional KVP

GetLegendGraphic	XML and optional KVP

8 GetCapabilities operation (mandatory)

8.1 General

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

8.2 GetCapabilities operation request

The GetCapabilities operation request shall be as specified in Subclauses 7.2 and 7.3 of [OGC 06-121r3]. The value of the "service" parameter shall be "WVS". The allowed set of service metadata (or Capabilities) XML document section names and meanings shall be as specified in Tables 6 and 10 of [OGC 06-121r3], with the additions listed in Table 13 below.

Table 13 — Additional Section name values and meanings

Section name	Meaning
PortrayalCapabilities	Return "PortrayalCapabilities" section in service metadata document

The "Multiplicity and use" column in Table 3 of [OGC 06-121r3] specifies the optionality of each listed parameter in the GetCapabilities operation request. Table 14 specifies the implementation of those parameters by WVS clients and servers.

Table 14 — Implementation of parameters in GetCapabilities operation request

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

acceptFormats AcceptFormats	Zero or one (optional)	If parameter provided, shall allow default or specified response	default response If parameter implemented and received, shall provide specified response

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

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

```
http://hostname:port/path?SERVICE=WVS&REQUEST=GetCapabilities&ACCEPTVER
SIONS=0.6.0&SECTIONS=Contents&UPDATESEQUENCE=XYZ123&ACCEPTFORMATS=text/
xml
```

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

```
<?xml version="1.0" encoding="UTF-8"?>
<WVSGetCapabilities xmlns="http://www.opengis.net/wvs/0.6.0"
    xmlns:ows="http://www.opengis.net/ows/1.1"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.opengis.net/wvs/0.6.0
../wvsGetCapabilities.xsd"
    service="WVS" request="GetCapabilities" updateSequence="XYZ123">
    <AcceptVersions>
        <ows:Version>0.6.0</ows:Version>
    </AcceptVersions>
    <Sections>
        <ows:Section>Contents</ows:Section>
    </Sections>
    <AcceptFormats>
        <ows:OutputFormat>text/xml</ows:OutputFormat>
    </AcceptFormats>
</WVSGetCapabilities>
```

NOTE An example for a complete GetCapabilities request can be found in [OGC 06-121r3] clause 7.2.4.

8.3 GetCapabilities operation response

8.3.1 Normal response

The service metadata document shall contain the WVS sections specified in Table 15. Depending on the values in the Sections parameter of the GetCapabilities operation request, any combination of these sections can be requested and shall be returned when requested.

Table 15 — Section name values and contents

Section name	Contents
ServiceIdentification	Metadata about this specific server. The schema of this section shall be the same as for all OWSs, as specified in Subclause 7.4.4 and owsServiceIdentification.xsd of [OGC 06-121r3].
ServiceProvider	Metadata about the organization operating this server. The schema of this section shall be the same for all OWSs, as specified in Subclause 7.4.5 and owsServiceProvider.xsd of [OGC 06-121r3].
OperationsMetadata	Metadata about the operations specified by this service and implemented by this server, including the URLs for operation requests. The basic contents and organization of this section shall be the same as for all OWSs, as specified in Subclause 7.4.6 and owsOperationsMetadata.xsd of [OGC 06-121r3].
Contents	Metadata about the data served by this server. For the WVS, this section shall contain data about layers, as specified in Subclause 8.3.3 below.
PortrayalCapabilities	Metadata about the PortrayalCapabilities served by this server. For the WVS, this section shall contain information about available image layers, available projections, available image styles, hints for near and far plane and meaningful viewpoints, as specified in Subclause 8.3.4 below.

In addition to these sections, each service metadata document shall include the mandatory "version" and optional "updateSequence" parameters specified in Table 9 in Subclause 7.4.2 of [OGC 06-121r3].

8.3.2 OperationsMetadata section standard contents

For the WVS, the OperationsMetadata section shall contain all parameters as specified in Subclause 7.4.6 and owsOperationsMetadata.xsd of [OGC 06-121r3]. The mandatory values of various (XML) attributes shall be as specified in Table 16. Similarly, the optional attribute values listed in Table 17 shall be included or not depending on whether that operation is implemented by that server. In Table 16 and Table 17, the "Attribute name" column uses dot-separator notation to identify parts of a parent item. The "Attribute value**"** column references an operation parameter, in this case an operation name, and the meaning of including that value is listed in the right column.

Table 16 — Required values of OperationsMetadata section attributes

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

Table 17 — Optional values of OperationsMetadata section attributes

Attribute name	Attribute value	Meaning of attribute value

Operation.name	GetFeatureInfo	The GetFeatureInfo operation is implemented by this server
	GetIdentifierMapping	The GetIdentifierMapping operation is implemented by this server.
	GetPosition	The GetPosition operation is implemented by this server
	GetMeasurement	The GetMeasurement operation is implemented by this server
	GetCamera	The GetCamera operation is implemented by this server
	GetLayerInfo	The GetLayerInfo operation is implemented by this server
	GetLegendGraphic	The GetLegendGraphic operation is implemented by this server

In addition to the optional values listed in Table 17, there are many optional values of the "name" attributes and "value" elements in the OperationsMetadata section, which may be included when considered useful. Most of these attributes and elements are for recording the domains of various parameters and quantities.

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

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

8.3.2.1 HTTP POST Encoding

All WVS servers shall specify the encodings that may be sent using HTTP POST transfer of operation requests. Specifically, an ows:Constraint element shall be included, with "PostEncoding" as the value of the "name" attribute and specifying different allowed values for each allowed encoding:

a) The value "SOAP" shall indicate that SOAP encoding is allowed when using HTTP POST transfer. The value "SOAP" is only valid for "PostEncoding".

b) The value "XML" shall indicate that XML encoding is allowed (without SOAP message encapsulation).

c) The value "KVP" shall indicate that KVP encoding is allowed, when using HTTP POST transfer.

If the HTTP POST connect point URL is different for different encodings of the operation requests, this ows:Constraint element shall be included in each Post element. If the connect point URL is the same for all encodings of all operation requests, this ows:Constraint element shall be included in the OperationsMetadata element.

8.3.2.2 Parameter values

All WVS servers shall specify the possible encodings of the responses for each supported operation as specified in Table 16 and Table 17 if additional encodings to the default encoding are available. Specifically, ows:Parameter elements shall be included, with the parameter name in the "name" attribute and the allowed encodings as list of ows:Value elements. If ows:Parameter elements are included, the default encodings shall be included in the list.

For each of the GetView and (if supported) GetLegendGraphic operations, an ows:Parameter element with name “ExceptionFormat” shall specify possible formats in which a client can be notified of service exceptions, if these operations support other exception formats than “XML”. Other possible values are “INIMAGE” and “BLANK” (see clause 7.2.5). The default value for all WVS operations is “XML”, encoded as text/xml.

Example for Parameter elements of the GetView operation:

```
<ows:Parameter name="ExceptionFormat">
   <ows:AllowedValues>
      <ows:Value>XML</ows:Value>
      <ows:Value>INIMAGE</ows:Value>
      <ows:Value>BLANK</ows:Value>
   </ows:AllowedValues>
</ows:Parameter>
```

COMMENT Shall the available output formats of all operations be advertised by ows:Parameter elements?

8.3.3 Contents section contents

The Contents section of a service metadata document contains metadata about the data served by this server. For the WVS, this Contents section shall contain data about the name, title, bounding box, and supported CRSs for each layer, and information about available backgrounds. The Contents section shall include the parameters specified in Table 18 through Table 23.

NOTE The following tables have been partly copied from W3DS draft specification [9], WPVS draft spec [7], and OWS Common [OGC 06-121r3].

Table 18 — Parts included in Contents section

Names	Definition	Data type and values	Multiplicity and use
Layer	Metadata describing a data set available from this server	Layer data structure, see Table 19	Zero or more (optional) Include as much as layers shall be advertised
Background	Metadata describing a background available for a GetView request	Background data structure, see Table 23	Zero or more (optional) Include when backgrounds shall be advertised

Table 19 — Parts of Layer data structure

Names [aa]	Definition	Data type and values	Multiplicity and use
title Title	Title of this dataset, normally used for display to a human	Language String data structure, see [OGC 06-121r3] clause 10.7	One (mandatory)
abstract Abstract	Brief narrative description of this dataset, normally available for display to a human	Language String data structure, see [OGC 06-121r3] clause 10.7	Zero or one (optional) Include when server chooses, recommended and usually included
keywords Keywords	Unordered list of one or more commonly used or formalized word(s) or phrase(s) used to describe this dataset	See MD_Keywords class in ISO 19115	Zero or more (optional) One for each keyword authority used
identifier Identifier	Unambiguous identifier of this dataset, unique for this WVS server	Character String type, not empty	Zero or one (optional) Include when may need to reference this dataset
wgs84BoundingBox WGS84BoundingBox	Minimum bounding rectangle surrounding this dataset, using WGS 84 CRS with decimal degrees and longitude before latitude	WGS84BoundingBoxType see Subclause 10.2 of [OGC 06-121r3]	Zero or one (optional) Include when useful or needed [a, b]
boundingBox [ff] BoundingBox	BoundingBox surrounding all or part of this dataset, in different CRS [c]	BoundingBoxType, see Subclause 10.2 of [OGC 06-121r3]	Zero or more (optional) Include when relevant and available [a, d]
metadata Metadata	Reference to more metadata about this layer	ows:Metadata, see [OGC 06-121r3] Table 32	Zero or more (optional) Include when useful
datasetSummary DatasetSummary	Metadatadescribing one subsidiary dataset available from this server	Layer data structure, see this Table	Zero or more (optional) One for each subsidiary Layer
availableCRS AvailableCRS	Coordinate reference system in which from this dataset may be requested	URI Unordered list	Zero or more (optional) Include when needed [e]
availableStyle AvailableStyle	Identifier of pre-defined style available for this Layer	Style data structure, see Table 20	Zero or more (optional) Include when relevant and available
minScaleDenominator MinScaleDenominator	Approximate minimum map scale denominator for useful client display	PositiveNumber type TBD	Zero or one (optional) Include when useful, recommended
maxScaleDenominator MaxScaleDenominator	Maximum scale denominator at which it is useful to display data of this layer	PositiveNumber type TBD	Zero or one (optional) Include when useful, recommended

Names [aa]	Definition	Data type and values	Multiplicity and use
startTime StartTime	Earliest data and time at which features are available	Date type, not empty, Data type defined in ISO 8601:2004(E)	Zero or one (optional) Include when useful
endTime EndTime	Latest date and time at which features are available	Date type, not empty, Data type defined in ISO 8601:2004(E)	Zero or one (optional) Include when useful
DataSources	Provides metadata about data sources used by Layers	DataSources data structure, see Table 21	Zero or one (optional) Include when available and useful

aa Although some values listed in the “Name” column appear to contain spaces, they shall not contain spaces.

bb Software may display the “Identifier” value when the “Title” is absent

ff There is no provision for describing disjoint bounding boxes like in WFS

a If this parameter is not recorded for a dataset, any parameter value recorded for a higher level in a hierarchy of datasets shall apply to this dataset. If this parameter is recorded for a dataset, any parameter value recorded for a higher level in a hierarchy of datasets shall be ignored.

b For each lowest-level dataset in a hierarchy, at least one applicable WGS84BoundingBox shall be recorded. If multiple WGS84 bounding boxes are included, this shall be interpreted as the union of the areas of these bounding boxes.

c More generally, definition of the horizontal, vertical, and temporal extent of this specific dataset. This parameter is included in addition to WGS84BoundingBox to allow more precise specification of the area in AvailableCRSs different from the WGS 84 CRS with decimal degrees and longitude before latitude.

d These Bounding Boxes shall not use any CRS not listed as an AvailableCRS. However, an AvailableCRS may be listed without a corresponding Bounding Box. If multiple bounding boxes are included with the same CRS, this shall be interpreted as the union of the areas of these bounding boxes.

e For each lowest-level dataset in a hierarchy, at least one applicable AvailableCRS shall be recorded.

f Any AvailableFormat recorded for a higher level in a hierarchy of datasets shall also apply to this dataset (TBR). For each lowest-level dataset in a hierarchy, at least one applicable AvailableFormat shall be recorded.

COMMENT So far not included: Format, Dimension data structure, opaque, noSubset, fixedWidth, fixedHeight, ElevationModel

Table 20 — Parts of Style data structure

Names	Definition	Data type and values	Multiplicity and use

Names	Definition	Data type and values	Multiplicity and use
title Title	Title of this style, human readable	LanguageString type, see [OGC 06-121r3] clause 10.7	Zero or one (optional) Include one for each language represented, recommended
abstract Abstract	Brief narrative description of this style, normally available for display to a human	LanguageString type, see [OGC 06-121r3] clause 10.7	Zero or one (optional) Include one for each language represented
keywords Keywords	Unordered list of one or more commonly used or formalized word(s) or phrase(s) used to describe this style	See MD_Keywords class in ISO 19115	Zero or one (optional) One for each keyword authority used
identifier Identifier	Unambiguous identifier of this style, unique for the containing layer	ows:CodeType	One (mandatory)
legendURL LegendURL	URL of legend for this style	LegendURL (TBD, from WMS 1.3.0)	Zero or more (optional) Include for each TBD
styleSheetURL StyleSheetURL	TBD	StyleSheetURL (TBD, from WMS 1.3.0)	Zero or one (optional) Include when TBD
styleURL StyleURL	TBD	StyleURL (TBD, from WMS 1.3.0)	Zero or one (optional) Include when TBD
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

Table 21 — Parts of DataSources data structure

Names	Definition	Data type and values	Multiplicity and use
dataProvider DataProvider	Reference to provider of Layer or collection of Layers, useful for display to user	DataProvider data structure, see Table 22	Zero or one (optional) Include when useful and available
datasetReference DatasetReference	Reference to electronically accessible dataset	URL	Zero or one (optional) Include when complete Layer may be retrieved
featureList FeatureList	Reference to electronically accessible feature list in dataset data	URL	Zero or one (optional) Include when feature list may be retrieved
cascaded Cascaded	Indicates whether the data of this layer was obtained from another server	Integer type, not empty	Zero or one (optional) Include when required
queryable Queryable	Indicates whether the server supports the GetFeatureInfo and GetLayerInfo operations on this layer	Boolean type, not empty	Zero or one (optional) Include when Layer shall be queryable

Table 22 — Parts of DataProvider data structure

Names	Definition	Data type and values	Multiplicity and use
providerName ProviderName	Name of data provider, human readable	CharacterString type, not empty	Zero or one (optional) Include when useful
providerSite ProviderSite	Reference to most relevant web site of data provider	See CI_OnlineResource class in ISO 19115	Zero or one (optional) Include when useful
logoURL LogoURL	URL of logo image	LogoURL data type (TBD, see WMS 1.3.0)	Zero or one (optional) Include when available and useful

Table 23 — Parts of Background data structure

Names	Definition	Data type and values	Multiplicity and use
title Title	Title of this background, normally used for display to a human	Language String data structure, see [OGC 06-121r3] clause 10.7	One (mandatory) [bb]
abstract Abstract	Brief narrative description of this background, normally available for display to a human	Language String data structure, see [OGC 06-121r3] clause 10.7	Zero or one (optional) Include when server chooses, recommended and usually included
keywords Keywords	Unordered list of one or more commonly used or formalized word(s) or phrase(s) used to describe this background	See MD_Keywords class in ISO 19115	Zero or more (optional) One for each keyword authority used
identifier Identifier	Unambiguous identifier of this background, unique for this WVS server	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 background	ows:Metadata, see [OGC 06-121r3] Table 32	Zero or more (optional) Include when useful

NOTE The UML class diagram contained in Subclause C.5 provides a useful graphical view of the contents of the Contents section listed in Table 18 – Table 23.

The “Multiplicity and use” columns in Table 6 through Table 16 in [OGC 06-121r3], and in Table 4, Table 18, and Table 19 of this document, specify the optionality of each listed parameter and data structure in the GetCapabilities operation response. All the “mandatory” parameters and data structures shall be implemented by all OWS servers, using a specified value(s).

The “updateSequence” parameter defined in Table 6 of [OGC 06-121r3] is optional implementation by OWS servers. As indicated in Table 14 of this document, the “updateSequence” parameter may be implemented by each server, but a specific OWS is allowed to require or prohibit server implementation of this parameter. If a specific OWS requires server implementation of this parameter, this parameter shall also be required in the operation response. Similarly, if a specific OWS prohibits server implementation of this parameter, this parameter shall also be prohibited in the operation response.

All other “optional” parameters and data structures, in the GetCapabilities operation response, should be implemented by all OWS servers using specified values, whenever and wherever each is considered useful metadata for that server.

8.3.4 PortrayalCapabilities section contents

The PortrayalCapabilities section of a service metadata document contains metadata about available ImageLayers and Formats, Projections, and ImageStyles served by this server. Additionally it gives hints near plane, far plane, and suggests viewpoints. The PortrayalCapabilities section shall be structured as specified in Table 24 through Table 30.

Table 24 — Parts of PortrayalCapabilities section

Names	Definition	Data type and values	Multiplicity and use
availableImageLayer AvailableImageLayer	Image layer that can be served by this server	ImageLayer data type, see Table 30 At least default color layer shall be supported	One or more (mandatory) One for each provided image layer
availableProjection AvailableProjection	Advertising a projection that is supported by this server	Projection data type, see Table 26	One or more (mandatory) Include one for each supported projection
nearPlaneHint NearPlaneHint	Hint at convenient near plane parameter value	PositiveNumber type	One (Mandatory)
farPlaneHint FarPlaneHint	Hint at convenient far plane parameter value	PositiveNumber type	One (Mandatory)
availableImageStyle AvailableImageStyle	Style to be applied to the overall 3D scene	ows:BasicIdentification type, see [OGC 06-121r3] clause 10.6.4	Zero or more (optaionl) Include one for each image style
viewpointHint ViewpointHint	Suggestion for good camera positions	Viewpoint data type, see Table 30	Zero or more (optional) Include when useful
supportsMultipleViews SupportsMultipleViews	Signals if a WVS server supports the retreival of multiple views or image layers	Boolen type Default shall be true	Zero or one (optional) Include when mutlipart response is not supported
availableNavigationType AvailableNavigationType	Names and describes the supported smart navigation types	NavigationType data type, see Table 29	Zero or more (optional) Include if GetCamera operation supported
supportedSpatialSelections	Signals if a WVS server supports spatial	List of ows:CodeType, not empty.	Zero or One (optional) Include if spatial

Names	Definition	Data type and values	Multiplicity and use
SupportedSpatialSelections	selection and lists the supported spatial selection methods	Allowed values of CodeList type: "overlaps", "contains_center", "cut"	selection is supported

Table 25 — Parts of ImageLayer data structure

Names	Definition	Data type and values	Multiplicity and use
title Title	Title of this image layer type, normally used for display to a human	Language String data structure, see [OGC 06-121r3] clause 10.7	Zero or one (optional) Include when desired
abstract Abstract	Brief narrative description of this image layer type, normally available for display to a human	Language String data structure, see [OGC 06-121r3] clause 10.7	Zero or one (optional) Include when desired
keywords Keywords	Unordered list of one or more commonly used or formalized word(s) or phrase(s) used to describe this image layer type	See MD_Keywords class in ISO 19115	Zero or more (optional) One for each keyword authority used
identifier Identifier	Unambiguous identifier of this ImageLayer, unique for this WVS server	CodeList type, not empty	One (mandatory)
AvailableFormat	Formats that are available for this ImageLayer	ows:MIME type, not empty, see [OGC 06-121r3] clause 10.5	One or more (mandatory) Include one for each supported image layer format

Table 26 — Parts of Projection data structure

Names	Definition	Data type and values	Multiplicity and use
projectionType ProjectionType	Identifier of the projection type, unique for this server	Character String type, not empty	One (mandatory)
projectionParameter ProjectionParameter	Parameters that specify this projection	DomainType data type, see [OGC 06-121r3] clause 13.2.1 This data type has an attribute described in Table 27	Zero or more (optional) Include one for each parameter of the projection

Table 27 — Parts of ProjectionParameter data structure

Names	Definition	Data type and values	Multiplicity and use

Names	Definition	Data type and values	Multiplicity and use
required Required	Flag, signaling if projection parameter is required of can be omitted	Boolean type Default shall be "true"	Zero or one (optional) Include when other than default "true" desired

Table 28 — Parts of Viewpoint data structure

Names	Definition	Data type and values	Multiplicity and use
title Title	Title of this background, normally used for display to a human	Language String data structure, see [OGC 06-121r3] clause 10.7	One (mandatory)
abstract Abstract	Brief narrative description of this viewpoint, normally available for display to a human	Language String data structure, see [OGC 06-121r3] clause 10.7	Zero or one (optional) Include when server chooses, recommended and usually included
keywords Keywords	Unordered list of one or more commonly used or formalized word(s) or phrase(s) used to describe this background	See MD_Keywords class in ISO 19115	Zero or more (optional) One for each keyword authority used
identifier Identifier	Unambiguous identifier of this viewpoint, unique for this WVS server	Character String type, not empty	Zero or one (optional) Include when may need to reference this dataset
projection Projection	Camera and projection specification for generating images and retrieving scene-related data	ProjectionBaseType E.g., PerspectiveProjection | OrthographicProjection, see Table 8 and Table 9	Zero or more Not each operation request includes Projection data structure, others shall contain exactly one or more

Table 29 — Parts of NavigatonType data structure

Names	Definition	Data type and values	Multiplicity and use

Names	Definition	Data type and values	Multiplicity and use
title Title	Title of this navigation type	Language String data structure, see [OGC 06-121r3] clause 10.7	One (mandatory) [bb]
abstract Abstract	Brief narrative description of this navigation type	Language String data structure, see [OGC 06-121r3] clause 10.7	Zero or one (optional) Include when server chooses, recommended and usually included
keywords Keywords	Unordered list of one or more commonly used or formalized word(s) or phrase(s) used to describe this navigation type	See MD_Keywords class in ISO 19115	Zero or more (optional) One for each keyword authority used
identifier Identifier	Unambiguous identifier of this navigation type, unique for this WVS server	Character String type, not empty	Zero or one (optional) Include when may need to reference this dataset
minPositions MinPosisitions	Minimal number of desired 2D positions, in Image CS	Number type Default is “1”	Zero or one (optional) Include when minimum number other than 1 desired
maxPositions MaxPositions	Maximum number of 2D positions, in Image CS	Number type, Default is undefined number	Zero or one (optional) Include when maximum number shall be restricted

NOTE The UML class diagram contained in Subclause C.4 provides a useful graphical view of the contents of the PortrayalCapabilities section listed in Table 24 – Table 30.

The “Multiplicity and use” columns in Table 6 through Table 16 in [OGC 06-121r3], and in Table 4, Table 18, and Table 19 of this document, specify the optionality of each listed parameter and data structure in the GetCapabilities operation response. All the “mandatory” parameters and data structures shall be implemented by all OWS servers, using a specified value(s).

The “updateSequence” parameter defined in Table 6 of [OGC 06-121r3] is optional implementation by OWS servers. As indicated in Table 14 of this document, the “updateSequence” parameter may be implemented by each server, but a specific OWS is allowed to require or prohibit server implementation of this parameter. If a specific OWS requires server implementation of this parameter, this parameter shall also be required in the operation response. Similarly, if a specific OWS prohibits server implementation of this parameter, this parameter shall also be prohibited in the operation response.

All other “optional” parameters and data structures, in the GetCapabilities operation response, should be implemented by all OWS servers using specified values, whenever and wherever each is considered useful metadata for that server.

8.3.4.1 AvailableImageLayer

Table 30 lists the identifiers and optionalities of the image layers the WVS currently specifies (see clause 6.5). However, a WVS server can implement and advertise additional image layers. For each supported image layer, an element <AvailableImageLayer> shall be included in the server’s metadata containing the image layer identifier.

Table 30 — ImageLayers specified by the WVS

Image Layer Identifier	Optionality	Description
COLOR	Mandatory	Image showing the colored 3D view. Default value
OBJECTID	Optional	Image containing color-encoded unique object ids
DEPTH	Optional	Image containing color-encoded distance from POC to surface points
NORMAL	Optional	Image containing color-encoded surface’s normal value
MASK	Optional	Image containing only binary no-data or data values per pixel.

8.3.4.2 AvailableProjection

Each Projection parameter specifies a projection type that is supported by the WVS server, the projection parameters and their possible values. Out of the In a WVS request (e.g., GetView) one ore more projections have to be requested, based on the projection type name in the server metadata (Table 8 and Table 9).

8.3.4.3 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 WVS shall give hints on NearPlaneHint and FarPlaneHint in the server’s metadata.

8.3.4.4 AvailableImageStyle

For styling the overall image a WVS GetView request can contain image styles. All the available image styles shall be advertised in the server’s metadata.

8.3.4.5 ViewpointHint

A WVS server can advertise one or more Viewpoint specifications, which can be used for generating views that show specific features of areas of the 3D scene. Viewpoints contain identification information and one projection. See clause 7.2.4 for available Projection types and their parameters. The projection specification can be used in a GetView request by a WVS client to generate the corresponding 3D view.

NOTE The Title and Abstract parameters of a ViewpointHint can be used for allowing a human user to choose from the predefined viewpoints in a WVS client.

8.3.4.6 SupportsMultipleViews

This flag signals, if a WVS server supports responding multiple views or multiple image layers by a single GetView response as HTTP multipart/mixed message (see clause 9.3.2).

8.3.4.7 AvailableNavigationType

A WVS can support navigation in the presented 3D scene by the GetCamera operation. Based on a set of 2D image pixels, a WVS would compute a new Projection that can be used with a subsequent GetView request. The AvailableNavigatonType element describes a navigation type and specifies how many 2D positions are required.

8.3.5 Capabilities document XML encoding

A XML schema fragment for a WVS service metadata document extends ows:CapabilitiesBaseType in owsCommon.xsd of [OGC 06-121r3], and is:

```
<!-- ============================================================= -->
<element name="WVS_Capabilities" type="wvs:CapabilitiesType"/>
<!-- ============================================================= -->
<complexType name="CapabilitiesType">
	<complexContent>
		<extension base="ows:CapabilitiesBaseType">
			<sequence>
				<element ref="wvs:Contents" minOccurs="1"/>
				<element ref="wvs:PortrayalCapabilities" minOccurs="0" maxOccurs="1"/>
			</sequence>
		</extension>
	</complexContent>
</complexType>
<!-- ============================================================= -->
<element name="Contents" type="wvs:ContentsType"/>
<!-- ============================================================= -->
<complexType name="ContentsType">
	<complexContent>
		<extension base="ows:ContentsBaseType">
			<sequence>
				<element name="Background" type="wvs:BackgroundType" minOccurs="0" maxOccurs="unbounded"/>
				<element name="Layer" type="wvs:LayerType" minOccurs="0" maxOccurs="unbounded"/>
			</sequence>
		</extension>
	</complexContent>
</complexType>
<!-- =========================================================== -->
<element name="PortrayalCapabilities" type="wvs:PortrayalCapabilitiesType"/>
<!-- =========================================================== -->
<complexType name="PortrayalCapabilitiesType">
	<annotation>
		<documentation>Describes the portrayal capabilities.</documentation>
	</annotation>
	<sequence>
		<element name="AvailableImageLayer" type="wvs:ImageLayerType" minOccurs="0" maxOccurs="unbounded"/>
		<element name="AvailableProjection" type="wvs:ProjectionType" minOccurs="1" maxOccurs="unbounded"/>
		<element name="NearPlaneHint" type="double" minOccurs="1"/>
		<element name="FarPlaneHint" type="double" minOccurs="1"/>
```

```
          <element name="AvailableImageStyle" type="wvs:ImageStyleType" minOccurs="0" maxOccurs="unbounded"/>
          <element name="ViewpointHint" type="wvs:ViewpointType" minOccurs="0" maxOccurs="unbounded"/>
          <element name="SupportsMultipleViews" type="boolean" minOccurs="0" default="false"/>
          <element name="SupportedSpatialSelections" minOccurs="0">
               <complexType>
                    <sequence>
                         <element name="Method" type="ows:CodeType" minOccurs="1" maxOccurs="3"/>
                    </sequence>
               </complexType>
          </element>
     </sequence>
</complexType>
```

As indicated, this XML Schema Document uses the owsServiceIdentification.xsd, owsServiceProvider.xsd, and owsOperationsMetadata.xsd schemas specified in [OGC 06-121r3]. It also uses an XML Schema Document for the “Contents” section of the WVS Capabilities XML document, which shall be as attached in the wvsGetCapabilities.xsd file. 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 06-121r3].

8.3.6 Capabilities document example

In response to GetCapabilities operation request, a WVS server might generate a document that looks like that in Annex D.

8.3.7 Exceptions

When a WVS server encounters an error while performing a GetCapabilities operation, it shall return an exception report message as specified in Clause 8 of [OGC 06-121r3]. The allowed exception codes shall include those listed in Table 5 of Subclause 7.4.1 of [OGC 06-121r3], if the updateSequence parameter is implemented by the server.

9 GetView operation (mandatory)

9.1 General

The GetView operation allows a WVS client to retrieve 3D views on a 3D scene managed and rendered by the WVS. Upon receiving a GetView request, a WVS shall either satisfy the request or issue a service exception.

Besides color images showing the 3D view, the GetView operation allows to retrieve additional geometric and thematic data as additional image layers. The supported image layers are described in 6.5.

Additionally, multiple 3D views and/or multiple images layers can be requested with a single GetView request. For each PortrayalOutput (e.g., camera specification and projection) that is specified, all requested image layers are generated according to the requested output formats and responded as parts of a single MIME multipart/mixed message.

In addition to styling features and the overall visual appearance, the background of a responded image can be styled.

9.2 GetView operation request

9.2.1 GetView request parameters

A request to perform the GetView operation shall use the data structure specified in Table 31 and Table 32. This table also specifies the UML model data type, source of values, and multiplicity of each listed parameter, plus the meaning to servers when each optional parameter is not included in the operation request. Although some values listed in the “Names” column appear to contain spaces, they shall not contain spaces.

NOTE 1 To reduce the need for readers to refer to other documents, the first three parameters listed below are largely copied from Table 26 in Subclause 9.2.1 of [OGC 06-121r3].

Table 31 — Parameters in GetView operation request

Names	Definition	Data type and values	Multiplicity and use
service service	Service type identifier	Character String type, not empty Value shall be "WVS"	One (mandatory)
request request	Operation name	Character String type, not empty Value shall be "GetView"	One (mandatory)
version version	Standard version for operation	Character String type, not empty Value is specified by each Implementation Standard and Schemas version	One (mandatory)
crs CRS	CRS to apply to BoundingBox and Projection parameters [a]	URI [b]	One (mandatory)
dataSelection DataSelection	Selection of data to portray	DataSelection type See Table 5	One (mandatory)
dataStyling DataStyling	Styling of the selected data	DataStyling type See Table 7	One (mandatory)
background Background	Identifier of desired background [e]	Character String type, not empty Values are specified in service metadata	Zero or one (optional) Include when background desired
backgroundColor BackgroundColor	Background color desired [e]	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
transparentBackground TransparentBackground	Background transparency desired [e]	CharacterString type, not empty Value either "true", "false". 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, see Table 32	One (mandatory)
exceptions Exceptions	Reference to format in which operation exceptions should be returned	CodeList type, either: "XML", "INIMAGE", "BLANK" Default is "XML"	Zero or one (optional) Include when exception format other than "XML" desired

a In general, this CRS may be 2D horizontal, 3D horizontal plus vertical, or 4D horizontal plus vertical plus time. A 4D CRS shall be a compound CRS, and a 3D CRS may be a compound CRS.

b The CRS shall be specified using either the European Survey Group form “EPSG:<POSC Code>” or the URL format defined OWS Common. Examples: “EPSG:31467”, “urn:ogc:def:crs:EPSG:6.3:31467”, see [OGC 06-121r3] clause 10.3 and [OGC 04-046r3].

c The maximum value of Width and Height may be specified using the Parameter element in the Operation element in the OperationsMetadata of the server’s metadata (Capabilities).

e 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.

f If no BoundingBox parameter is given, a WVS server shall ignore a given SpatialSelection parameter.

Table 32 — Parameters in 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 type, see Table 8 and Table 9, 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 06-121r3] 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.

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

NOTE 3 The UML class diagram contained in Subclause C.6 provides a useful graphical view of the contents of the GetView operation request listed in Table 31 and Table 32.

COMMENT Is there a better name for parameter ImageLayers (ImageDimensions, …)?

COMMENT Should “Formats” be renamed to “OutputFormats”?

The “Multiplicity and use” columns in Table 31 through Table 32 specify the optionality of each listed parameter and data structure in the GetView operation request. All the “mandatory” parameters and data structures shall be implemented by all WVS clients, using a specified value(s). Similarly, all the “mandatory” parameters and data structures shall be implemented by all WVS servers, checking that each request parameter or data structure is received with any specified value(s).

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

9.2.1.1 ImageLayers

The mandatory ImageLayer parameter specifies a list of image layers to retrieve. A WVS shall specify the supported image layers in the server’s metadata document (see 8.3.4.1).

9.2.1.2 Background

The optional parameter Background can be used to include a predefined background advertised in the server’s metadata. The value of the Background parameter refers to an existing background identifier of the server. If no Background parameter is provided with the GetView request the server uses the BackgroundColor parameter for coloring the background. If both are not provided, the WVS server decides which background to include.

9.2.1.3 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 WVS shall set the background pixels to the color specified by the BackgroundColor parameter.

A WVS server 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.

9.2.1.4 Transparent

The optional Transparent 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 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 less common. The image/jpeg format does not provide transparency at all.

When Transparent is set to true and the Formats parameter contains an image format (e.g. image/gif), then a WVS 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 server 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 WVS server shall generate a transparent background only, when no Background and BackgroundColor parameter are provided with the GetView request.

9.2.1.5 Portrayals

The mandatory Portrayals parameter specifies one or multiple portrayals to be generated by a WVS. It contains one or multiple PortrayalOutput as value, which specify image size (width and height), projections, output formats, and image qualities (see Table 32).

9.2.1.5.1 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 6.9.2) 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.

9.2.1.5.2 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. So far, the WVS specifies PerspectiveProjection and OrthographicProjection, see Table 8 and Table 9.

9.2.1.5.3 Formats

The mandatory Formats parameter specifies a list of image formats as ows:MimeType, see [OGC 06-121r3] 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 server's metadata (9.2.1.1).

9.2.1.5.4 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 Quality parameter is only useful for COLOR images; for other image layer types, the compression algorithms can generate invalid data. Additionally, the Qualitative parameter is only useful for output formats that support image compression. If a requested output format does not support different qualities, the WVS shall ignore the parameter for this format.

9.2.1.6 Exceptions

The optional Exceptions parameter states the format in which to report errors during processing a GetView request as one of "XML" (default), "INIMAGE", or "BLANK" (see clause 7.2.5).

A WVS server shall accept Exceptions formats other than "XML" only, if it is requested for at least one COLOR ImageLayer in any image format, e.g., image/jpeg. A WVS client should request Exception formats other than "XML" only if it requests at least one COLOR ImageLayer encoded as image type, and the responded image is dedicated for being displayed.

9.2.2 GetView request KVP encoding (mandatory)

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 33. The parameters listed in Table 33 shall be as specified in Table 31 and Table 32 above.

Table 33 — GetView operation request URL parameters

Name and example [a]	Optionality and use	Definition and format
service=WVS	Mandatory	Service type identifier
request= GetView	Mandatory	Operation name
version=0.6.0	Mandatory	Standard and schema version for this operation

crs=EPSG:26916	Mandatory	CRS to apply to BoundingBox and Viewpoint(s)
boundingbox=202759.0,3310170.0,30.0,213200.0,3320896.0,100.0	Optional, include when spatial selection by bounding box desired	Bounding box surrounding desired subset of layer(s), in desired CRS
spatialselection=contains_center	Optional, include when spatial selection desired	Indicates method of selecting objects with BoundingBox
time=2009-11-11T11:11:00	Optional, include when spatial selection other than “current” desired	Date and time
layers=dem,bldgs	Mandatory	Identifier(s) of desired data layer(s)
styles=default,default	Optional, include when named layer style desired	Identifier(s) of desired layer style(s)
sld=http://myhost/ myBldgStyle.xml	Optional, mutally exclusive with SLD_Body [c]	URL reference to SLD document
sld_body=TBD	Optional, mutally exclusive with SLD [c]	
imagestyles=Foggy,Illustrative	Optional, include when image style desired	Identifier(s) of desired overall image style(s)
background=skybox	Optional, include when background desired	Identifier of desired background
backgroundcolor=0xFF0000	Optional, include when background color desired	Background color desired
transparentbackground=true	Optional, include when transparent background desired	Non-data parts shall be transparent
Portrayals=WIDTH=1024;HEIGHT=1024;Projections=Perspective,202000,3310000,200,202000,3305000,200,0,0,1,60,,0.01,1000.0;IMAGELAYERS=COLOR,DEPTH;FORMATS=image/jpeg,image/png%3Bmode=32bit;QUALITIES=100,100@WIDTH=768;HEIGHT=768;Projections=Perspective,202000,3310000,200,202000,3305000,200,0,0,1,60,,0.01,1000.0;IMAGELAYERS=COLOR,DEPTH;FORMATS=image/png,image/png; QUALITIES=100,100 [b]	Mandatory	List of specifications of Width (one), Height (one), Projections (one or more), ImageLayers (one or more), Formats (one or more), Quality (one or more)
Exceptions=INIMAGE	Optional, include when default XML not desired	Format of exceptions

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 06-121r3].

b The value for this field shall be encoded as specified in section 9.2.2.2

c Either sld or sld_body shall be used. TBD: “layers” parameter should become optional

EXAMPLE An example GetView operation request KVP encoded for HTTP GET is:

```
http://hostname:port/path?SERVICE=WVS&VERSION=0.6.0&REQUEST=GetView&CRS
=EPSG:26916&BOUNDINGBOX=202759.0,3310170.0,30.0,213200.0,3320896.0,100.
0&SPATIALSELECTION=contains_center&LAYERS=all&STYLES=&IMAGESTYLES=Foggy
```

```
&BACKGROUND=skybox&PORTRAYALS=WIDTH=1024;HEIGHT=1024;Projections=Perspe
ctive,210000,332000,50,211000,332000,30,0,0,1,60,,0.01,1000.0;IMAGELAYE
RS=COLOR;FORMATS=image/jpeg;QUALITIES=85&EXCEPTIONS=INIMAGE
```

9.2.2.1 BoundingBox

The value of the BoundingBox parameter is a list of comma-separated real numbers as described [OGC 06-121r3] 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.

EXAMPLE A 3D BoundingBox is KVP encoded in the form "minx,miny,minz,maxx,maxy,maxz".

9.2.2.2 Layers

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

9.2.2.3 Styles

The mandatory Styles parameter shall be a comma-separated list of Style Identifiers as advertised in the server'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.

9.2.2.4 Encoding of Portrayals values (mandatory)

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.

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 [IETF RFC 1738].

9.2.2.4.1 GetView Portrayals parameter KVP encoding

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};
ProjectionParameterValue = empty | ‘URL encoded value’
ProjectionTypeName =   “Perspective” | “Orthographic” |
                       ‘name of other supported projection’;

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)};

Value = ‘URL encoded value’;
```

URL encoding 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”.

9.2.2.4.2 Projections parameter KVP encoding

The Projections parameter (which is a parameter of the PortrayalOutput type) shall be encoded as comma separated list of tuple value. 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 8 and Table 9) the values can be empty. In this case the server has to use default values, ignore parameters, or compute values.

The Projections parameter shall contain multiple projections only, when the WVS server advertised the capability to generate multiple view and/or image layers by the optional <SupportsMultipleViews> element in the server metadata.

9.2.3 GetView request XML encoding (optional)

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

```
<!-- ============================================================= -->
<element name="GetView" type="wvs:GetViewType"/>
<!-- ============================================================= -->
<complexType name="GetViewType">
	<annotation>
		<documentation>XML encoded WVS GetView operation request.  </documentation>
	</annotation>
	<complexContent>
		<extension base="wvs:WVSRequestBaseType">
			<sequence>
				<element name="CRS" type="anyURI"/>
				<element ref="ows:BoundingBox" minOccurs="0"/>
				<element name="SpatialSelection" type="ows:CodeType" minOccurs="0"/>
				<element name="Time" type="dateTime" minOccurs="0"/>
				<element name="Layers" type="wvs:IdentifierListType" minOccurs="1"/>
				<element name="Styles" type="wvs:IdentifierListType" minOccurs="0"/>
				<element name="ImageStyles" type="wvs: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="wvs: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"?>
<wvs:GetView
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xmlns:wvs="http://www.opengis.net/wvs/0.6.0"
	xsi:schemaLocation="http://www.opengis.net/wvs/0.6.0 ../wvsGetView.xsd"
	xmlns:ows="http://www.opengis.net/ows/1.1"
	xmlns:xlink="http://www.w3.org/1999/xlink"
	service="WVS" request="GetView" version="0.6.0">
	<wvs:CRS>EPSG:1234</wvs: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>
	<wvs:SpatialSelection>contains_center</wvs:SpatialSelection>
	<wvs:Time>2009-11-11T11:11:00</wvs:Time>
	<wvs:Layers>
		<ows:Identifier>dem</ows:Identifier>
		<ows:Identifier>bldgs</ows:Identifier>
	</wvs:Layers>
	<wvs:Styles>
		<ows:Identifier>nice</ows:Identifier>
		<ows:Identifier></ows:Identifier>
	</wvs:Styles>
	<wvs:ImageStyles>
		<ows:Identifier>illustrative</ows:Identifier>
		<ows:Identifier>npr</ows:Identifier>
	</wvs:ImageStyles>
	<wvs:Background>skybox</wvs:Background>
```

```
    <wvs:BackgroundColor>0xFF0000</wvs:BackgroundColor>
    <wvs:Exceptions>INIMAGE</wvs:Exceptions>
    <wvs:Portrayals>
        <wvs:PortrayalOutput>
            <wvs:Width>1024</wvs:Width>
            <wvs:Height>1024</wvs:Height>
            <wvs:Projections>
                <wvs:PerspectiveProjection>
                    <wvs:POC>100 100 100</wvs:POC>
                    <wvs:POI>101 101 101</wvs:POI>
                    <wvs:Up>0 0 1</wvs:Up>
                    <wvs:FOVX>80</wvs:FOVX>
                    <wvs:FOVY>60</wvs:FOVY>
                    <wvs:NearPlane>0.0001</wvs:NearPlane>
                    <wvs:FarPlane>10000.0</wvs:FarPlane>
                </wvs:PerspectiveProjection>
                <wvs:OrthographicProjection>
                    <wvs:POC>100 100 100</wvs:POC>
                    <wvs:POI>101 101 101</wvs:POI>
                    <wvs:Up>0 0 1</wvs:Up>
                    <wvs:Left>-100</wvs:Left>
                    <wvs:Right>-100</wvs:Right>
                    <wvs:Bottom>-100</wvs:Bottom>
                    <wvs:Top>100</wvs:Top>
                    <wvs:NearPlane>0.0001</wvs:NearPlane>
                    <wvs:FarPlane>10000.0</wvs:FarPlane>
                </wvs:OrthographicProjection>
            </wvs:Projections>
            <wvs:ImageLayers>
                <ows:Identifier>COLOR</ows:Identifier>
                <ows:Identifier>DEPTH</ows:Identifier>
                <ows:Identifier>OBJECTID</ows:Identifier>
            </wvs:ImageLayers>
            <wvs:Formats>
                <wvs:Format>image/jpeg</wvs:Format>
                <wvs:Format>image/png; mode=32bit</wvs:Format>
                <wvs:Format>text/xml</wvs:Format>
            </wvs:Formats>
            <wvs:Qualities>80,,</wvs:Qualities>
        </wvs:PortrayalOutput>
    </wvs:Portrayals>
</wvs:GetView>
```

9.3 GetView operation 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.

9.3.1 Normal response XML encoding

COMMENT Optionally a WVS server could advertise XML encoding for image layers.

9.3.2 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 [IETF 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 “WVS_MULTIPART_MESSAGE_BOUNDARY”, which is also indicated in the Content-Type header of the multipart response message.

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 34 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 34 — 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)

NOTE 1 Nested multipart messages are possible, i.e., one message part can contain a multipart message itself.

NOTE 2 According to the [IETF RFC 2046] a multipart message also ends with the message’s boundary string.

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

A WVS consumer that requests multiple image layers shall be capable to parse the multipart response and to extract the message parts.

9.3.3 GetView exceptions

When a WVS server encounters an error while performing a GetView operation, it shall return an exception report message as specified in Clause 8 of [OGC 06-121r3]. The allowed standard exception codes shall include those listed in Table 35. For each listed exceptionCode, the contents of the “locator” parameter value shall be as specified in the right column of Table 35.

NOTE To reduce the need for readers to refer to other documents, the first four values listed below are copied from Table 25 in Subclause 8.3 of [OGC 06-121r3].

NOTE This table is mainly copied from W3DS draft

Table 35 — Exception codes for GetView operation

exceptionCode value	Meaning of code	“locator” value
OperationNotSupported	Request is for an operation that is not supported by this server	Name of operation not supported
MissingParameterValue	Operation request does not include a parameter value, and this server did not declare a default value for that parameter	Name of missing parameter
InvalidParameterValue	Operation request contains an invalid parameter value	Name of parameter with invalid value
OptionNotSupported	Request is for an option that is not supported by this server	Identifier of option not supported
CRSNotSupported	Operation request contains a value in the CRS parameter which is not supported by the server	Name of unsupported CRS
InvalidListLength	Operation request contains Style or LOD parameter whose value does not contain a list of the same length as in parameter Layers	Name of parameter with invalid value
UnknownLayer	Operation request contains an identifier in the Layers parameter which is unknown to the server	Identifier of invalid layer
UnknownStyle	Operation request contains an identifier in the Style parameter which is unknown to the server	Identifier of invalid style
FormatNotSupported	Operation request contains a MIME type in the Format parameter which is not supported by the server	Name of unsupported format
ExceptionNotSupported	Operation request contains a value in the Exception parameter which is not supported by the server	Name of unsupported exception format
SpatialSelectionNotSupported	Operation request contains a value in BoundingBox or SpatialSelection, but spatial selection is not supported at all	“BoundingBox” or “SpatialSelection”
InvalidProjection	Projection is invalid, e.g., POC equals POI, or UP vector is collinear with view direction	Name of invalid Projection parameter
NoApplicableCode	No other exceptionCode specified by this service and server applies to this exception	None, omit “locator” parameter

10 GetFeatureInfo operation (optional)

10.1 General

The GetFeatureInfo operation is designed to provide clients per pixel with additional attribute information about features within a view that is currently displayed.

The canonical use case for GetFeatureInfo is that a user explores the response of a GetView request and points at an object within the view for which to obtain more information. The concept of this operation is that the client determines the 2D position a user clicked at and submits this location together with additional parameters to the server. Since the WVS protocol is stateless, most of the GetView parameters that were used for generating the view need to be submitted so that the WVS is able to reconstruct that view. 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.

A WVS can provide various formats for structuring and responding feature attribute information. However, the format of the response must be provided as MIME type, e.g. “text/xml; subtype=gml/3.1.1”, enabling the client to parse the result properly. The WVS defines “text/xml” as a default format and specifies a schema for responding feature information in a table-like structure.

Conceptually, a WVS server performs a ray intersection test for determining the features on the ray from POC to the selected (and projected) position. Due to 3D space, a single feature could be hit several times by this ray. However, a single feature shall be included at most once in a GetFeatureInfo operation response. Additionally, multiple features of the same layer could be hit; because of that, the number of responded features per layer can be restricted.

The GetFeatureInfo operation is only supported for those Layers for which the attribute queryable="1" (true) has been defined or inherited. A client shall not issue a GetFeatureInfo request for other layers. A WVS shall respond with a properly formatted service exception response (code = OperationNotSupported) if it receives a GetFeatureInfo request but does not support it.

COMMENT Do we need an Info_Layers parameter besides the Layers parameter from GetView request?

10.2 GetFeatureInfo operation request

10.2.1 GetFeatureInfo request parameters

A request to perform the GetFeatureInfo operation shall use the data structure specified in Table 36. This table also specifies the UML model data type, source of values, and multiplicity of each listed parameter, plus the meaning to servers when each optional parameter is not included in the operation request. Although some values listed in the “Names” column appear to contain spaces, they shall not contain spaces.

NOTE 1 To reduce the need for readers to refer to other documents, the first three parameters listed below are largely copied from Table 26 in Subclause 9.2.1 of [OGC 06-121r3].

Table 36 — Parameters in GetFeatureInfo operation request

Names [a]	Definition	Data type and values	Multiplicity and use
service service	Service type identifier	Character String type, not empty Value shall be “WVS”	One (mandatory)
request request	Operation name	Character String type, not empty Value shall be “GetFeatureInfo”	One (mandatory)
version version	Standard version for operation	Character String type, not empty Value is specified by each Implementation Standard and Schemas version	One (mandatory)
crs CRS	CRS to apply to BoundingBox and Projection parameters [a]	URI [b]	One (mandatory)
dataSelection DataSelection	Data selection and subset selection specifying the data to retrieve data from [a]	DataSelection type See Table 5	One (mandatory)
dataStyling DataStyling	Layer-specific styling, feature-specific styling, and overall view styling	DataStyling type See Table 7	One (mandatory)
width Width	Width of output image that the request refers to, in pixels	PositiveInteger type [c]	One (mandatory)
height Height	Height of output image that the request refers to, in pixels	PositiveInteger type [c]	One (mandatory)
projection Projection	Camera specification and projection parameters	Projection type, see Table 8 and Table 9 Supported Projections are specified in service metadata	One (mandatory)
position Position	Pixel position for which to search for features to return information from	Characters String type, comma-separated list of X and Y coordinates, in image CS	One (mandatory)
featureCount FeatureCount	Number of features to return information from	Integer type, not zero Default = 1	Zero or one (optional) Include when more than one feature information per layer desired
format Format	Return format for feature information	MIME type, see [OGC 06-121r3] clause 10.5 Values are specified in server metadata. Default is “text/xml”	Zero or one (mandatory) Include when format other than “text/xml” desired

GetView RequestPart

a The Layers parameter in DataSelection shall only include layers that are specified as “queryable” in the Contents section of the server’s metadata.

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

NOTE 3 The UML class diagram contained in Subclause C.7 provides a useful graphical view of the contents of the GetFeatureInfo operation request listed in Table 36.

The “Multiplicity and use” columns in Table 36 specify the optionality of each listed parameter and data structure in the GetFeatureInfo operation request. All the “mandatory” parameters and data structures shall be implemented by all WVS clients, using a specified value(s). Similarly, all the “mandatory” parameters and data structures shall be implemented by all WVS servers, checking that each request parameter or data structure is received with any specified value(s).

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

10.2.1.1 Layers parameter in DataSelection

The mandatory Layers parameter specifies a list of those data layers for which the server selects features based on the pixel position and collects attribute information from. Each entry in the Layers list shall refer to a layer identifier as described in the server’s metadata; these Layer shall have the attribute queryable=”1” (true) defined or inherited.

10.2.1.2 Format

The mandatory Format parameter specifies the target encoding of the returned attribute information provided as ows:MimeType, see [OGC 06-121r3] clause 10.5. Available formats are described in the server’s metadata. Default format is “text/xml”.

10.2.1.3 FeatureCount

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

10.2.1.4 Position

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.

10.2.2 GetFeatureInfo request KVP encoding (optional)

Servers may implement HTTP GET transfer of the GetFeatureInfo operation request, using KVP encoding. The KVP encoding of the GetFeatureInfo operation request shall use the parameters specified in Table 37. The parameters listed in Table 37 shall be as specified in Table 43 above.

Table 37 — GetFeatureInfo operation request URL parameters

Name and example [a]	Optionality and use	Definition and format
service=WVS	Mandatory	Service type identifier
request=GetFeatureInfo	Mandatory	Operation name
version=0.6.0	Mandatory	Standard and schema version for this operation
TBD: GetViewRequestPart [b]	Mandatory	Partial copy of the request parameters that generated the view for which information is desired
featurecount=5	Optional	Number of features to return information from (default=1)
position=271,127	Mandatory	Pixel position for which to search for features to return information from
format=text/xml	Mandatory	Format encoding of the result

a All parameter names are here listed using mostly lower case letters. However, any parameter name capitalization shall be allowed in KVP encoding, see Subclause 11.5.2 of [OGC 06-121r3].

b The GetViewRequestPart includes the Layers parameter specifying layers to retrieve the data from.

EXAMPLE An example GetFeatureInfo operation request KVP encoded for HTTP GET is:

```
http://hostname:port/path?SERVICE=WVS&REQUEST=GetFeatureInfo&VERSION=0.
6.0&GETVIEWREQUESTPART&LAYERS=bldg&FORMAT=text/xml+;subtype%3Dgml/3.1.1
s&FEATURECOUNT=5&POSITION=271,127
```

TBD: The GetViewRequestPart element still has to be substituted by the corresponding fields: DataSelection, DataStyling, Width, Height, and Projection as shown in the Table 36.

10.2.3 GetFeatureInfo request XML encoding (mandatory)

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

```
<complexType name="GetFeatureInfoType">
	<annotation>
		<documentation>XML encoded WVS GetFeatureInfo operation request. </documentation>
	</annotation>
	<complexContent>
		<extension base="wvs:WVSRequestBaseType">
			<sequence>
				<element ref="wvs:GetViewRequestPart" />
				<element name="FeatureCount" type="integer" minOccurs="0"/>
				<element name="Position" type="wvs:Position2DType"/>
				<element name="Format" type="ows:MimeType"/>
	</sequence>
```

```
        </extension>
    </complexContent>
</complexType>
```

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

```
<?xml version="1.0" encoding="UTF-8"?>
<GetFeatureInfo xmlns="http://www.opengis.net/wvs/0.6.0"
    xmlns:ows="http://www.opengis.net/ows/1.1"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.opengis.net/wvs/0.6.0 ../wvsGetFeatureInfo.xsd"
    service="WVS" version="0.6.0" request="">
    <GetViewRequestPart />
    <FeatureCount>1</FeatureCount>
    <Position>100 200</Position>
    <Format>text/xml</Format>
</GetFeatureInfo>
```

10.3 GetFeatureInfo operation response

10.3.1 Normal response parameters

The normal response to a valid GetFeatureInfo operation request shall be a table or equivalent data structure in which the attribute information of each found feature is listed. The response is a document encoded in the MIME type as specified in the Format parameter. More precisely, a response from the GetFeatureInfo operation shall include the parts listed in Table 38. This table also specifies the UML model data type plus the multiplicity and use of each listed part.

According to the default Format parameter “text/xml” the normal response is XML encoded. This XML response could be easily transformed into other representations, e.g., HTML by applying XSL transformation.

NOTE Feature information could also be provided in more specific format such as, e.g., GML or CityGML, if implemented by the WVS server and if advertised in the server’s metadata.

Table 38 — Parts of GetFeatureInfo operation response

Names	Definition	Data type and values	Multiplicity and use
featureInfo FeatureInfo	Base XML element	FeatureInfo data type	One (mandatory)

Table 39 — Parts of FeatureInfo data structure

Names	Definition	Data type and values	Multiplicity and use
featureInfoList FeatureInfoList	List containing feature information for all features found in one data layer One for each requested layer for that features where found	FeatureInfoList type	One or more (mandatory) Include one for each data layer for that features were identified and feature information retrieved

Table 40 — Parts of FeatureInfoList data structure

Names	Definition	Data type and values	Multiplicity and use
typeName TypeName	Name of the feature type, that are found in that data layer	Character String type, not empty	One (mandatory)
featureAttributeList FeatureAttributeList	List of feature attributes One for each feature	List of FeatureAttribute data type, not empty	One or more (mandatory) One for each found feature

Table 41 — Parts of FeatureAttributeList data structure

Names	Definition	Data type and values	Multiplicity and use
Attribute	Feature attribute value	FeatureAtribute data type extending Character String type (see Table 42)	One or more (mandatory) One for each attribute of the specific feature

Table 42 — Parts of FeatureAttribute data structure

Names	Definition	Data type and values	Multiplicity and use
name name	Name of FeatureAttribute	CharacterString type	One (mandatory)

NOTE The UML class diagram contained in Subclause C.7 provides a graphical view of the contents of the GetFeatureInfo operation response listed in Table 38 - Table 42.

10.3.2 Normal response XML encoding

The following schema fragment specifies the contents and structure of a GetFeatureInfo operation response, always encoded in XML, when the request parameter Format is “text/xml”:

```
<!-- ============================================================ -->
<element name="FeatureAttributeList" type="wvs:FeatureAttributeListType"/>
<!-- ============================================================ -->
<complexType name="FeatureAttributeListType">
	<sequence>
		<element name="Attribute" type="wvs:FeatureAttributeType" minOccurs="0" maxOccurs="unbounded">
		</element>
	</sequence>
</complexType>
<!-- ============================================================ -->
<complexType name="FeatureAttributeType">
	<simpleContent>
		<extension base="string">
			<attribute name="name" type="string" use="required">
				<annotation>
					<documentation>Name of the attribute.</documentation>
				</annotation>
			</attribute>
		</extension>
	</simpleContent>
</complexType>
```

10.3.3 GetFeatureInfo response example

A GetFeatureInfo operation response for WVS can look like this encoded in text/xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<FeatureInfo xmlns="http://www.opengis.net/wvs/0.6.0"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.opengis.net/wvs/0.6.0 ../wvsGetFeatureInfo.xsd">
        <FeatureInfoList>
                <TypeName>Building</TypeName>
                <FeatureAttributeList>
                        <Attribute name="id"></Attribute>
                        <Attribute name="objkey">1123 Hochschulgebaeude</Attribute>
                        <Attribute name="strkey">2390</Attribute>
                        <Attribute name="hsno">1</Attribute>
                        <Attribute name="description">Alte Universitaet</Attribute>
                        <Attribute name="sockelhoehe">2.500000</Attribute>
                        <Attribute name="traufhoehe">15.00000</Attribute>
                        <Attribute name="height">115.347270</Attribute>
                </FeatureAttributeList>
        </FeatureInfoList>
</FeatureInfo>
```

10.3.4 GetFeatureInfo exceptions

When a WVS server encounters an error while performing a GetFeatureInfo operation, it shall return an exception report message as specified in Clause 8 of [OGC 06-121r3]. The allowed standard exception codes shall include those listed in Table 43. For each listed exceptionCode, the contents of the “locator” parameter value shall be as specified in the right column of Table 43.

NOTE To reduce the need for readers to refer to other documents, the first four values listed below are copied from Table 25 in Subclause 8.3 of [OGC 06-121r3].

Table 43 — Exception codes for GetFeatureInfo operation

exceptionCode value	Meaning of code	“locator” value
OperationNotSupported	Request is for an operation that is not supported by this server	Name of operation not supported
MissingParameterValue	Operation request does not include a parameter value, and this server did not declare a default value for that parameter	Name of missing parameter
InvalidParameterValue	Operation request contains an invalid parameter value	Name of parameter with invalid value
OptionNotSupported	Request is for an option that is not supported by this server	Identifier of option not supported
CRSNotSupported	Operation request contains a value in the CRS parameter which is not supported by the server	Name of unsupported CRS
UnknownLayer	Operation request contains an identifier in the Layers parameter which is unknown to the server.	Identifier of invalid layer
FormatNotSupported	Operation request contains a MIME type in the Format parameter which is not supported by the server	Name of unsupported format
TBD	TBD: Exceptions of the GetView operation	TBD

NoApplicableCode	No other exceptionCode specified by this service and server applies to this exception	None, omit “locator” parameter

11 GetIdentifierMapping (optional)

11.1 General

The optional GetIdentifierMapping allows WVS clients to retrieve the mapping of object identifiers to feature identifiers based on

a) a list of object identifiers, which was, .e.g., retrieved from object id layer via GetView operation or

b) a list of feature identifiers.

Based on the retrieved information, a WVS consumer could request the original feature data from a data layer’s data source.

11.2 GetIdentifierMapping operation request

11.2.1 GetIdentifierMapping request parameters

A request to perform the GetMeasurement operation shall use the data structure specified in Table 44. This table also specifies the UML model data type, source of values, and multiplicity of each listed parameter, plus the meaning to servers when each optional parameter is not included in the operation request. Although some values listed in the “Names” column appear to contain spaces, they shall not contain spaces.

NOTE 1 To reduce the need for readers to refer to other documents, the first three parameters listed below are largely copied from Table 26 in Subclause 9.2.1 of [OGC 06-121r3].

Table 44 — Parameters in GetIdentifierMapping operation request

Names [a]	**Definition**	**Data type and values**	**Multiplicity and use**
service service	Service type identifier	Character String type, not empty Value shall be "WVS"	One (mandatory)
request request	Operation name	Character String type, not empty Value shall be "GetIdentifierMapping"	One (mandatory)
version version	Standard version for operation	Character String type, not empty Value is specified by each Implementation Standard and Schemas version	One (mandatory)
objectIds ObjectIds	List of object identifiers for which to retrieve feature identifiers [a]	List of Integer data type, can be empty	Zero or one (optional)
featureIds FeatureIds	List of feature identifiers for which to retrieve object identifiers [a]	List of Character String data type, can be empty	Zero or one (optional)
format Format	Format encoding of the result	ows:MimeType, see [OGC 06-121r3] clause 10.5	One (mandatory)

a At lest one of "ObjectIds" or "FeatureIds" parameters is required.

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

NOTE 3 The UML class diagram contained in Subclause C.8 provides a useful graphical view of the contents of the GetIdentifierMapping operation request listed in Table 56.

The "Multiplicity and use" columns in Table 44 specify the optionality of each listed parameter and data structure in the GetMeasurement operation request. All the "mandatory" parameters and data structures shall be implemented by all WVS clients, using a specified value(s). Similarly, all the "mandatory" parameters and data structures shall be implemented by all WVS servers, checking that each request parameter or data structure is received with any specified value(s).

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

11.2.1.1 ObjectIds, FeatureIds

The ObjectIds parameter specifies one or more object identifiers (e.g., retrieved from an OBJECTID layer) for which to retrieve the corresponding feature identifiers. The FeatureIds parameter specifies one or more feature identifiers for which to retrieve the corresponding object identifiers, which would be assigned to these features in an OBJECTID layer.

ObjectIds and/or FeatureIds parameters can be empty. In this case a WVS server shall respond all the available object/feature identifier mappings.

QUESTION Should the number of retrieved mappings be restricted?

One or both of the ObjectIds or FeatureIds parameter shall be provided by a WVS consumer. The server shall respond with an exception (locator=”Identifiers”) when a GetIdentifierMapping request does not contain one of the ObjectIds or FeatureIds parameters.

11.2.2 GetIdentifierMapping KVP encoding (optional)

Servers may implement HTTP GET transfer of the GetIdentifierMapping operation request, using KVP encoding. The KVP encoding of the GetIdentifierMapping operation request shall use the parameters specified in Table 45. The parameters listed in Table 45 shall be as specified in Table 44 above.

Table 45 — GetIdentifierMapping operation request URL parameters

Name and example [a]	Optionality and use	Definition and format
service=WVS	Mandatory	Service type identifier
request=GetIdentifierMapping	Mandatory	Operation name
version=0.6.0	Mandatory	Standard and schema version for this operation
objectids=1111,2222	Optional [b]	Object identifiers for which to retrieve the corresponding feature identifiers
featureids=fid2222,fid3333	Optional [b]	Feature identifiers for which to retrieve the corresponding object identifiers
format=text/xml	Mandatory	Format encoding of the result

a All parameter names are here listed using mostly lower case letters. However, any parameter name capitalization shall be allowed in KVP encoding, see Subclause 11.5.2 of [OGC 06-121r3].

b At least one of objectids or featureids parameters shall be provided.

EXAMPLE An example GetIdentifierMapping operation request KVP encoded for HTTP GET is:

```
http://hostname:port/path?SERVICE=WVS&VERSION=0.6.0&REQUEST=GetIdentifi
erMapping&OBJECTIDS=1111,2222&FEATUREIDS=fid2222,fid3333&FORMAT=text/xm
l
```

11.2.3 GetIdentifierMapping request XML encoding (mandatory)

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

```
<!-- =========================================================== -->
<element name="GetIdentifierMapping" type="wvs:GetIdentifierMappingType"/>
<!-- =========================================================== -->
<complexType name="GetIdentifierMappingType">
	<annotation>
		<documentation>XML encoded WVS GetIdentifierMapping operation request. </documentation>
	</annotation>
	<complexContent>
		<extension base="wvs:WVSRequestBaseType">
			<sequence>
				<element name="ObjectIds" minOccurs="0">
					<simpleType>
						<list itemType="integer"/>
					</simpleType>
				</element>
				<element name="FeatureIds" minOccurs="0">
					<simpleType>
						<list itemType="string"/>
					</simpleType>
				</element>
				<element name="Format" type="ows:MimeType"/>
			</sequence>
		</extension>
	</complexContent>
</complexType>
```

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

```
<?xml version="1.0" encoding="UTF-8"?>
<GetIdentifierMapping xmlns="http://www.opengis.net/wvs/0.6.0"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://www.opengis.net/wvs/0.6.0 ../wvsGetIdentifierMapping.xsd"
	service="WVS" version="0.6.0" request="GetIdentifier">
	<ObjectIds>1111 2222</ObjectIds>
	<FeatureIds>fid2222 fid3333</FeatureIds>
	<Format>text/xml</Format>
</GetIdentifierMapping>
```

11.3 GetIdentifierMapping operation response

11.3.1 Normal response parameters

The normal response to a valid GetIdentifierMapping operation request shall be a document containing

a) the mappings of object ids to feature ids and/or

b) the mapping of feature ids to object ids.

More precisely, a response from the GetMeasurement operation shall include the parts listed in Table 46. This table also specifies the UML model data type plus the multiplicity and use of each listed part.

Table 46 — Parts of GetIdentifierMapping operation response

Names	Definition	Data type and values	Multiplicity and use
identifierMappingResponse IdentifierMappingResponse	Base xml element	IdentifierMappingResponse type (Table 47)	One (mandatory)

Table 47 — Parts of IdentifierMappingResponse data structure

Names	Definition	Data type and values	Multiplicity and use
identifierMapping IdentifierMapping	Mapping for object identifier and feature identifier	IdentifierMapping data type [a]	One or more (mandatory) Include one for each determined mapping
a IdentifierMapping type is an empty type. Attributes are described in Table 48.			

Table 48 — Attributes of IdentifierMapping data structure

Names	Definition	Data type and values	Multiplicity and use
oid oid	Object identifier	String type, not empty	One (mandatory)
fid fid	Feature identifier	Character String type, not empty	One (mandatory)
layer layer	Identifier of the layer the feature belongs to	Character String type, not empty	One (mandatory)

NOTE The UML class diagram contained in Subclause C.8 provides a graphical view of the contents of the GetIdentifierMapping operation response listed in Table 46 - Table 48.

11.3.2 Normal response XML encoding

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

```
<!-- ============================================================= -->
<element name="IdentifierMappingResponse" type="wvs:IdentifierMappingResponseType"/>
<!-- ============================================================= -->
<complexType name="IdentifierMappingResponseType">
        <sequence>
                <element name="IdentifierMappings" minOccurs="0" maxOccurs="unbounded">
                        <complexType>
                                <sequence>
                                        <element ref="wvs:IdentifierMapping" minOccurs="1" maxOccurs="unbounded"/>
                                </sequence>
                        </complexType>
                </element>
        </sequence>
</complexType>
<!-- ============================================================= -->
<element name="IdentifierMapping" type="wvs:IdentifierMappingType"/>
<!-- ============================================================= -->
<complexType name="IdentifierMappingType">
```

```
        <attribute name="oid" use="required"/>
        <attribute name="fid" use="required"/>

        <attribute name="layer" use="required"/>
</complexType>
```

11.3.3 GetIdentifierMapping response example

A WVS operation response for GetIdentifierMapping can look like this encoded in XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<IdentifierMappingResponse xmlns="http://www.opengis.net/wvs/0.6.0"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.opengis.net/wvs/0.6.0 ../wvsGetIdentifierMapping.xsd">
        <IdentifierMappings>
        <IdentifierMapping oid="1111" fid="fid1111" layer="buildings"/>
        <IdentifierMapping oid="2222" fid="fid2222" layer="buildings"/>
        <IdentifierMapping oid="3333" fid="fid3333" layer="cityfurnitures"/>
        </IdentifierMappings>
</IdentifierMappingResponse>
```

11.3.4 GetIdentifierMapping exceptions

When a WVS server encounters an error while performing a GetIdentifierMapping operation, it shall return an exception report message as specified in Clause 8 of [OGC 06-121r3]. The allowed standard exception codes shall include those listed inTable 49. For each listed exceptionCode, the contents of the "locator" parameter value shall be as specified in the right column of Table 49.

NOTE To reduce the need for readers to refer to other documents, the first four values listed below are copied from Table 25 in Subclause 8.3 of [OGC 06-121r3].

Table 49 — Exception codes for GetIdentifierMapping operation

exceptionCode value	Meaning of code	"locator" value
OperationNotSupported	Request is for an operation that is not supported by this server	Name of operation not supported
MissingParameterValue	Operation request does not include a parameter value, and this server did not declare a default value for that parameter	Name of missing parameter
InvalidParameterValue	Operation request contains an invalid parameter value	Name of parameter with invalid value
OptionNotSupported	Request is for an option that is not supported by this server	Identifier of option not supported
TBD	TBD	TBD
NoApplicableCode	No other exceptionCode specified by this service and server applies to this exception	None, omit "locator" parameter

12 GetPosition operation (optional)

12.1 General

For a specific view, the optional GetPosition operation allows to compute and retrieve

a) the (nearest to the camera) 3D position at a specified 2D pixel of the 3D view, and/or

b) the 2D pixel that represents a specific 3D position.

Requested with a 2D pixel position, the server shall determine and respond the 3D position at the surface of the object that is visible at this pixel in the 3D view that was previously requested by a GetView request. Retrieving 3D positions allows in-image interaction, e.g., for selecting a point-of-interest, which can be used within a camera specification for generating a new view. The other away around, a client could request, if a specific 3D position, e.g., the last point of camera is visible in the image, and could highlight this position.

If a GetPosition request contains a 3D position, and this 3D position lies within the view frustum specified by the camera and projection specifications, a corresponding 2D pixel position shall be responded. Otherwise, i.e., the corresponding 3D position is not visible in the 3D view, the server shall

a) compute a pixel position outside the visible view, e.g., (-10,10), or

b) return empty coordinate values.

Using a 2D position retrieved from GetPosition (by using a 3D position as parameter) again for requesting GetPosition does not necessarily result in the original 3D position as the original 3D position could be occluded by other objects.

NOTE A 2D image pixel is rather covering an area of the scene than representing a specific point in the 3D scene. Thus, a 3D position retrieved by a GetPosition request will include a computational error.

A WVS server shall respond with a properly formatted service exception (XML) response (code = OperationNotSupported) if it receives a GetLayerInfo request but does not support it.

12.2 GetPosition operation request

A GetPosition request includes most of the parameters of a GetView request and, additionally, specifies 2D pixel positions and/or 3D positions respectively.

12.2.1 GetPosition request parameters

A request to perform the GetPosition operation shall use the data structure specified in Table 50. This table also specifies the UML model data type, source of values, and multiplicity of each listed parameter, plus the meaning to servers when each optional parameter is not included in the operation request. Although some values listed in the “Names” column appear to contain spaces, they shall not contain spaces.

NOTE 1 To reduce the need for readers to refer to other documents, the first three parameters listed below are largely copied from Table 26 in Subclause 9.2.1 of [OGC 06-121r3].

Table 50 — Parameters in GetPosition operation request

Names [a]	Definition	Data type and values	Multiplicity and use
service service	Service type identifier	Character String type, not empty Value shall be “WVS”	One (mandatory)
request request	Operation name	Character String type, not empty Value shall be “GetPosition”	One (mandatory)
version version	Standard version for operation	Character String type, not empty Value is specified by each Implementation Standard and Schemas version	One (mandatory)
crs CRS	CRS to apply to BoundingBox, Projection parameters, and 3D positions. [a]	URI [b]	One (mandatory)
dataSelection DataSelection	Data selection and subset selection specifying the data to retrieve data from [a]	DataSelection type See Table 5	One (mandatory)
dataStyling DataStyling	Layer-specific styling, feature-specific styling, and overall view styling	DataStyling type See Table 7	One (mandatory)
width Width	Width of output that image the request refers to, in pixels	PositiveInteger type [c]	One (mandatory)
height Height	Height of output image that the request refers to, in pixels	PositiveInteger type [c]	One (mandatory)
projection Projection	Camera specification and projection parameters	Projection type, see Table 8 and Table 9 Supported Projections are specified in service metadata	One (mandatory)
positions2D Positions2D	List of 2D positions for which to retrieve 3D position	List of CharacterString type, not empty List(s) of coordinate values of length 2, in image CS	Zero or one (optional) Include when requesting 2D position(s) desired
positions3D Positions3D	List of 3D position for which to retrieve pixel posisition	List of CharacterString type, not empty List(s) of coordinate values of length 3, in request CRS	Zero or one (optional) Include when requesting 3D position(s) desired
format Format	Format encoding of the result	ows:MimeType, see [OGC 06-121r3] clause 10.5	One (mandatory)

exceptions Exceptions	Format of exceptions	ows:MimeType, see [OGC 06-121r3] clause 10.5 Default is text/xml	Zero or one (optional) Include when other format than text/xml desired

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

NOTE 3 The UML class diagram contained in Subclause C.9 provides a useful graphical view of the contents of the GetPosition operation request listed in Table 50.

The "Multiplicity and use" columns in Table 50 specify the optionality of each listed parameter and data structure in the GetPosition operation request. All the "mandatory" parameters and data structures shall be implemented by all WVS clients, using a specified value(s). Similarly, all the "mandatory" parameters and data structures shall be implemented by all WVS servers, checking that each request parameter or data structure is received with any specified value(s).

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

12.2.1.1 Positions2D, Positions3D

The Positions2D parameter specifies one or more 2D positions for which to retrieve the corresponding 3D position. The Positions3D parameter specifies one or more 3D positions for which to retrieve the corresponding 2D positions in a view. 2D positions are specified as list of pairs of X and Y coordinates in image CS, i.e., starting from upper left corner = (0,0). 3D positions are specified as list of tuple of X, Y, and Z coordinates in request CRS.

One of the Positions2D or Positions3D parameter shall be provided by a WVS consumer. The server shall respond with an exception (locator="Positions") when a GetPosition request does not contain one of the Positions2D or Positions3D parameters.

12.2.1.2 Format

The mandatory Format parameter specifies the target encoding of the returned column and attribute information provided as ows:MimeType, see [OGC 06-121r3] clause 10.5. Available formats are described in the server's metadata.

12.2.2 GetPosition request KVP encoding (optional)

Servers may implement HTTP GET transfer of the GetPosition operation request, using KVP encoding. The KVP encoding of the GetPosition operation request shall use the parameters specified in Table 51. The parameters listed in Table 51 shall be as specified in Table 50 above.

Table 51 — GetPosition operation request URL parameters

Name and example [a]	Optionality and use	Definition and format
service=WVS	Mandatory	Service type identifier
request=GetPosition	Mandatory	Operation name
version=0.6.0	Mandatory	Standard and schema version for this operation
TBD: GetViewRequestPart	Mandatory	Partial copy of the request parameters that generated the view for which information is desired
positions2D=27,12,35,89	Optional [b]	2D position(s) in the image for which to retrieve 3D position(s) in the scene
positions3D=12343.0,543543.0,65.0	Optional [b]	3D position(s) in the 3D scene for which to retrieve 2D position(s) in the image
format=text/plain	Mandatory	Format encoding of the result

a All parameter names are here listed using mostly lower case letters. However, any parameter name capitalization shall be allowed in KVP encoding, see Subclause 11.5.2 of [OGC 06-121r3].

b At least one of Positions2D or Positions3D parameters shall be provided.

EXAMPLE An example GetPosition operation request KVP encoded for HTTP GET is:

```
http://hostname:port/path?SERVICE=WVS&VERSION=0.6.0&REQUEST=GetPosition
&GETVIEWREQUESTPART&POSITIONS2D=27,12,35,89&POSITIONS3D=12343,543543,65
&FORMAT=text/plain
```

12.2.3 GetPosition request XML encoding (mandatory)

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

```
<!-- ============================================================ -->
<element name="GetPosition" type="wvs:GetPositionType"/>
<!-- ============================================================ -->
<complexType name="GetPositionType">
	<annotation>
		<documentation>XML encoded WVS GetFeatureInfo operation request. </documentation>
	</annotation>
	<complexContent>
		<extension base="wvs:WVSRequestBaseType">
			<sequence>
				<<element ref="wvs:GetViewRequestPart" />
				<element name="Positions2D" type="wvs:Position2DListType" minOccurs="0"/>
				<element name="Positions3D" type="wvs:Position3DListType" minOccurs="0"/>
			</sequence>
		</extension>
	</complexContent>
</complexType>
```

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

```
<?xml version="1.0" encoding="UTF-8"?>
<GetPosition xmlns="http://www.opengis.net/wvs/0.6.0"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://www.opengis.net/wvs/0.6.0 ../wvsGetPosition.xsd" service="WVS" version="0.6.0"
request="">
	<GetViewRequestPart />
```

```
    <Positions2D>
        <Position>123 123</Position>
        <Position>9999999 9999999</Position>
    </Positions2D>
    <Positions3D>
        <Position>9999999 999999 -9999999</Position>
        <Position>21393.0 32133.0 46.0</Position>
    </Positions3D>
</GetPosition>
```

COMMENT Should the CRS be included in the responded positions?

12.3 GetPosition operation response

12.3.1 Normal response parameters

The normal response to a valid GetPosition operation request shall be a document containing for each requested Positions2D and Positions3D one 3D position or 2D position, respectively. More precisely, a response from the GetPosition operation shall include the parts listed in Table 52. This table also specifies the UML model data type plus the multiplicity and use of each listed part.

Position can be responded in plain text (text/plain) or XML encoded (text/xml).

If one position could not be determined, because a requested 3D position is not within the view frustum or because a requested 2D position does not hit any object surface, the according value shall be an undefined position.

Table 52 — Parts of GetPosition operation response

Names	Definition	Data type and values	Multiplicity and use
PositionResponse PositionResponse	Base xml element	PositionResponse data type	One (mandatory)

Table 53 — Parts of PositionResponse data structure

Names	Definition	Data type and values	Multiplicity and use
positions3D Positions3D	List of 3D positions	List of Position3D type, not empty [a]	Zero or one (optional) Include when 3D positions desired
positions2D Positions2D	List of 2D positions	List of Position2D type, not empty [a]	Zero or one (optional) Include when 2D positions desired
a The PositionResponse data structure shall contain at least one of Positions3D or Positions2D			

NOTE The UML class diagram contained in Subclause C.9 provides a graphical view of the contents of the GetPosition operation response listed in Table 52 and Table 53.

12.3.2 Normal response text encoding

For encoding as plain text, all coordinate values are concatenated, separated by commas; 3D positions before 2D positions. The order of the coordinates corresponds to the order of the corresponding 2D positions and 3D positions in the GetPosition request. Format for *m* returned 3D positions (3D) and *n* returned 2D positions (2D) is:

3D1X,3D1Y,3D1Z,…,3DmX,3DmY,3DmZ,2D1X,2D1Y,…,2DnX,2DnY

In text/plain encoding, undefined positions shall be encoded by empty coordinate values.

12.3.3 Normal response XML encoding

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

```
<!-- ============================================================ -->
<element name="PositionResponse" type="wvs:PositionResponseType"/>
<!-- ============================================================ -->
<complexType name="PositionResponseType">
	<sequence>
		<element name="Positions3D" type="wvs:Position3DListType" minOccurs="0"/>
		<element name="Positions2D" type="wvs:Position2DListType" minOccurs="0"/>
	</sequence>
</complexType>
```

In XML encoding, undefined <Position3D> and <Position2D> elements shall have attribute “nil” set ”true”.

12.3.4 GetPosition response example

A GetPosition operation response for WVS can look like this encoded in plain text (text/plain):

123456.44,12354.45,78.21,,,,,,124,421

A GetPosition operation response for WVS can look like this encoded in XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<PositionResponse xmlns="http://www.opengis.net/wvs/0.6.0"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://www.opengis.net/wvs/0.6.0 ../wvsGetPosition.xsd">
	<Positions3D>
		<Position xsi:nil="true"></Position>
		<Position>123456.44 12354.45 78.21</Position>
	</Positions3D>
	<Positions2D>
		<Position xsi:nil="true"></Position>
		<Position>124 421</Position>
	</Positions2D>
</PositionResponse>
```

12.3.5 GetPosition exceptions

When a WVS server encounters an error while performing a GetFeatureInfo operation, it shall return an exception report message as specified in Clause 8 of [OGC 06-121r3]. The allowed standard exception codes shall include those listed in Table 54. For each listed exceptionCode, the contents of the "locator" parameter value shall be as specified in the right column of Table 54.

NOTE To reduce the need for readers to refer to other documents, the first four values listed below are copied from Table 25 in Subclause 8.3 of [OGC 06-121r3].

Table 54 — Exception codes for GetPosition operation

exceptionCode value	Meaning of code	"locator" value
OperationNotSupported	Request is for an operation that is not supported by this server	Name of operation not supported
MissingParameterValue	Operation request does not include a parameter value, and this server did not declare a default value for that parameter	Name of missing parameter
InvalidParameterValue	Operation request contains an invalid parameter value	Name of parameter with invalid value
OptionNotSupported	Request is for an option that is not supported by this server	Identifier of option not supported
TBD	TBD	TBD
NoApplicableCode	No other exceptionCode specified by this service and server applies to this exception	None, omit "locator" parameter

13 GetMeasurement operation (optional)

13.1 General

The optional GetMeasurement operation provides capabilities for spatially analyzing a 3D view generated by a previous GetView request based on selected 2D pixel positions in that view.

The canonical use case for GetMeasurement is that a user explores the response of a GetView request, selects one of the analysis operations advertised by the WVS server, and selects pixel positions in the image. This data serves as parameters of the GetMeasurement operation. Based on that, the WVS server computes and responds a measurement result. Examples for analysis operations include distance measurement, area measurement, or visibility analysis.

A WVS server shall advertise measurement operations in its metadata. A WVS client can use the operation identifiers and descriptions for offering these functionalities to a service consumer. Various measurement operations can be provided by a WVS server. As a basic subset a WVS server that implements the GetMeasurement operation shall provide the measurement operations listed in Table 62.

Table 55 — Basic measurement operations provided by the WVS

Measurement operation Identifier	Use

Measurement operation Identifier	Use
Distance	Measuring Euclidean distances and path length between several positions.
Area	Measuring the area spanned by several positions.

13.1.1 Distance measurement

For each 2D pixel position (in a list of at least two 2D positions), a WVS server computes the corresponding 3D positions and computes the sum of the Euclidean distances between the consecutive 3D positions.

13.1.2 Area measurment

For each 2D pixel position (in a list of at least three 2D positions), a WVS server computes the corresponding 3D positions and computes the area of the surface spanned by these points. In 3D space, area measurement is not straight forward, as the derived 3D positions are not likely to be coplanar. Projecting the 3D positions onto a horizontal represents a forward solution to this problem. More accurate algorithms are possible, e.g., computing a best fitting plane for projection of applying differential technique to the problem. This standard does not specify any algorithms or accuracy measures. A WVS server could provide such information in its metadata.

13.1.3 Other measurement operations

Other measurement operations can be implemented, advertised, and provided by a WVS server. These operations shall be specified in the server’s metadata. Constraints such as maximum number of 2D positions shall be specified.

13.2 GetMeasurement operation request

13.2.1 GetMeasurement request parameters

A request to perform the GetMeasurement operation shall use the data structure specified in Table 56. This table also specifies the UML model data type, source of values, and multiplicity of each listed parameter, plus the meaning to servers when each optional parameter is not included in the operation request. Although some values listed in the “Names” column appear to contain spaces, they shall not contain spaces.

NOTE 1 To reduce the need for readers to refer to other documents, the first three parameters listed below are largely copied from Table 26 in Subclause 9.2.1 of [OGC 06-121r3].

Table 56 — Parameters in GetMeasurement operation request

Names [a]	Definition	Data type and values	Multiplicity and use
service service	Service type identifier	Character String type, not empty Value shall be "WVS"	One (mandatory)
request request	Operation name	Character String type, not empty Value shall be "GetMeasurement"	One (mandatory)
version version	Standard version for operation	Character String type, not empty Value is specified by each Implementation Standard and Schemas version	One (mandatory)
crs CRS	CRS to apply to BoundingBox and Projection parameters [a]	URI [b]	One (mandatory)
dataSelection DataSelection	Data selection and subset selection specifying the data to retrieve data from [a]	DataSelection type See Table 5	One (mandatory)
dataStyling DataStyling	Layer-specific styling, feature-specific styling, and overall view styling	DataStyling type See Table 7	One (mandatory)
width Width	Width of output image that the request refers to, in pixels	PositiveInteger type [c]	One (mandatory)
height Height	Height of output image that the request refers to, in pixels	PositiveInteger type [c]	One (mandatory)
projection Projection	Camera specification and projection parameters	Projection type, see Table 8 and Table 9 Supported Projections are specified in service metadata	One (mandatory)
operation Operation	Reference to the measurement operation to apply	Character String type, not empty Value is specified in the server meatadata	One (mandatory)
positions Positions	List of 2D positions to use as input for measurement operation	List of CharacterString type, not empty List of coordinate values of length 2, in image CS	Zero or one (optional) Include when 2D position(s) desired by measurement operation
format Format	Format encoding of the result	ows:MimeType, see [OGC 06-121r3] clause 10.5 Values are specified for each measurement operation in server metadata	One (mandatory)

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

NOTE 3 The UML class diagram contained in Subclause C.10 provides a useful graphical view of the contents of the GetMeasurement operation request listed in Table 56.

QUESTION Should the GetMeasurement operation allow for other input parameters than 2D positions?

The “Multiplicity and use” columns in Table 56 specify the optionality of each listed parameter and data structure in the GetMeasurement operation request. All the “mandatory” parameters and data structures shall be implemented by all WVS clients, using a specified value(s). Similarly, all the “mandatory” parameters and data structures shall be implemented by all WVS servers, checking that each request parameter or data structure is received with any specified value(s).

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

13.2.2 GetMeasurement request KVP encoding (optional)

Servers may implement HTTP GET transfer of the GetMeasurement operation request, using KVP encoding. The KVP encoding of the GetMeasurement operation request shall use the parameters specified in Table 56. The parameters listed in Table 57 shall be as specified in Table 56 above.

Table 57 — GetMeasurement operation request URL parameters

Name and example	Optionality and use	Definition and format
service=WVS	Mandatory	Service type identifier
request= GetMeasurement	Mandatory	Operation name
version=0.6.0	Mandatory	Standard and schema version for this operation
TBD: GetViewRequestPart	Mandatory	Partial copy of the request parameters that generated the view for which information is desired
operation=Distance	Mandatory	Reference to advertised measurement operation
positions=27,12,35,89	Optional	2D position(s) in the image for which to retrieve 3D position(s) in the scene
format=text/plain	Optional	Format encoding of the result

EXAMPLE An example GetMeasurement operation request KVP encoded for HTTP GET is:

```
http://hostname:port/path?SERVICE=WVS&VERSION=0.6.0&REQUEST=GetMeasurem
ent&GETVIEWREQUESTPART&OPERATION=Distance&POSITIONS=27,12,35,89&FORMAT=
text/plain
```

13.2.3 GetMeasurement request XML encoding (mandatory)

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

```
<!-- ========================================================= -->
<element name="GetMeasurement" type="wvs:GetMeasurementType"/>
<!-- ========================================================= -->
<complexType name="GetMeasurementType">
	<annotation>
		<documentation>XML encoded WVS GetFeatureInfo operation request. </documentation>
	</annotation>
	<complexContent>
		<extension base="wvs:WVSRequestBaseType">
			<sequence>
				<element ref="wvs:GetViewRequestPart"/>
				<element name="Operation" type="ows:CodeType" minOccurs="1"/>
				<element name="Positions2D" type="wvs:Position2DListType" minOccurs="0"/>
				<element name="Format" type="ows:MimeType"/>
			</sequence>
		</extension>
	</complexContent>
</complexType>
```

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

```
<?xml version="1.0" encoding="UTF-8"?>
<GetMeasurement xmlns="http://www.opengis.net/wvs/0.6.0"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://www.opengis.net/wvs/0.6.0 ../wvsGetMeasurement.xsd"
	service="WVS" version="0.6.0" request="">
	<GetViewRequestPart/>
	<Operation>Distance</Operation>
	<Positions2D>
		<Position>121 131</Position>
	</Positions2D>
	<Format>text/xml</Format>
</GetMeasurement>
```

13.3 GetMeasurement operation response

13.3.1 Normal response parameters

The normal response to a valid GetMeasurement operation request shall be a document containing the result of the requested measurement. More precisely, a response from the GetMeasurement operation shall include the parts listed in Table 65. This table also specifies the UML model data type plus the multiplicity and use of each listed part.

Measurement results can be responded in plain text (text/plain) or XML encoded (text/xml). For encoding as plain text, the response document contains the value and – space-separated- the unit of measure.

Table 58 — Parts of GetMeasurement operation response

Names	Definition	Data type and values	Multiplicity and use
MeasurementResponse MeasurmentResponse	Base xml element	MeasurementResponse type (Table 59)	One (mandatory)

Table 59 — Parts of MeasurementResponse data structure

Names	Definition	Data type and values	Multiplicity and use
result Result	Measurement result value	anyType type, contains attributes described in ResultData (Table 60)	One or more (mandatory) Include one for each desired result

Table 60 — Parts of Result data structure

Names	Definition	Data type and values	Multiplicity and use
dataType dataType	Data type of the result	URI type	One (mandatory)
uom uom	Unit of measure for result data	URI type	

NOTE The UML class diagram contained in Subclause C.10 provides a graphical view of the contents of the GetMeasurement operation response listed in Table 58 - Table 60.

QUESTION Should the GetMeasurement operation allow for more complex output such as ComplexData, LiteralData, BoundingBoxData in WPS?

13.3.2 Normal response XML encoding

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

```
<!-- ============================================================ -->
<element name="MeasurementResponse" type="wvs:MeasurementResponseType"/>
<!-- ============================================================ -->
<complexType name="MeasurementResponseType">
        <sequence>
                <element name="ResultData" type="wvs:ResultDataType" minOccurs="1" maxOccurs="unbounded"/>
        </sequence>
</complexType>
<!-- ============================================================ -->
<complexType name="ResultDataType">
        <complexContent>
                <extension base="anyType">
                        <attribute name="dataType" type="anyURI"/>
                        <attribute name="uom" type="anyURI"/>
                </extension>
        </complexContent>
</complexType>
```

13.3.3 GetMeasurement response example

A WVS operation response for GetMeasurement can look like this encoded in plain text (text/plain):

12345.54321 m

A GetMeasurement operation response for WVS can look like this encoded in XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<MeasurementResponse xmlns="http://www.opengis.net/wvs/0.6.0"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.opengis.net/wvs/0.6.0 ../wvsGetMeasurement.xsd">
        <Result dataType="integer" uom="meter">12345.54321</Result>
</MeasurementResponse>
```

13.3.4 GetMeasurement exceptions

When a WVS server encounters an error while performing a GetMeasurement operation, it shall return an exception report message as specified in Clause 8 of [OGC 06-121r3]. The allowed standard exception codes shall include those listed inTable 61. For each listed exceptionCode, the contents of the "locator" parameter value shall be as specified in the right column of Table 61.

NOTE To reduce the need for readers to refer to other documents, the first four values listed below are copied from Table 25 in Subclause 8.3 of [OGC 06-121r3].

Table 61 — Exception codes for GetMeasurement operation

exceptionCode value	Meaning of code	"locator" value
OperationNotSupported	Request is for an operation that is not supported by this server	Name of operation not supported
MissingParameterValue	Operation request does not include a parameter value, and this server did not declare a default value for that parameter	Name of missing parameter
InvalidParameterValue	Operation request contains an invalid parameter value	Name of parameter with invalid value
OptionNotSupported	Request is for an option that is not supported by this server	Identifier of option not supported
TBD	TBD	TBD
NoApplicableCode	No other exceptionCode specified by this service and server applies to this exception	None, omit "locator" parameter

14 GetCamera operation (optional)

COMMENT Should this operation be renamed (GetProjection,…)?

14.1 General

The optional GetCamera operation provides a general mechanism for supporting the navigation within the scene represented in a specific 3D view. For a set of 2D pixel positions in a specific view, the GetCamera operation computes a new camera specification, which is dedicated for retrieving a new, “good” view, e.g., on specific objects, in a subsequent GetView operation request. By implementing the GetCamera operation, a WVS server can support “smart” navigation techniques to simple clients.

As an example, a WVS server could compute “good” views regarding the selection of scene objects. Requesting the GetCamera operation with a selected point on a building façade, could result in a camera specification that –applied with a subsequent GetView request– leads to a close-up view of this building.

Semantics of underlying 3D geodata, user characteristics, or user tasks require different “smart” navigation techniques for a 3DGeoVE. A WVS server implementing the GetCamera operation shall advertise the supported navigation types and the number of 2D positions required as parameter in the server’s metadata. A default technique shall be provided and advertised.

The GetCamera operation shall describe the “good” view as Projection that can be easily used for a subsequent GetView request, e.g., as <PerspectiveProjection> element in the case of text/xml as output format. The projection specification shall be described in the same Projection type used for the initial GetView request.

14.2 GetCamera operation request

14.2.1 GetCamera request parameters

A request to perform the GetCamera operation shall include the data structure specified in Table 62. This table also specifies the UML model data type, source of values, and multiplicity of each listed parameter, plus the meaning to servers when each optional parameter is not included in the operation request. Although some values listed in the “Names” column appear to contain spaces, they shall not contain spaces.

NOTE 1 To reduce the need for readers to refer to other documents, the first three parameters listed below are largely copied from Table 26 in Subclause 9.2.1 of [OGC 06-121r3].

Table 62 — Parameters in GetCamera operation request

Names	Definition	Data type and values	Multiplicity and use
service service	Service type identifier	Character String type, not empty Value shall be “WVS”	One (mandatory)
request request	Operation name	Character String type, not empty Value shall be “GetCamera”	One (mandatory)
version version	Standard version for operation	Character String type, not empty Value is specified by each Implementation Standard and Schemas version	One (mandatory)
crs CRS	CRS to apply to BoundingBox and Projection parameters [a]	URI [b]	One (mandatory)
dataSelection DataSelection	Data selection and subset selection specifying the data to retrieve data from [a]	DataSelection type See Table 5	One (mandatory)
dataStyling DataStyling	Layer-specific styling, feature-specific styling, and overall view styling	DataStyling type See Table 7	One (mandatory)
width Width	Width of output image that the request refers to, in pixels	PositiveInteger type [c]	One (mandatory)
height Height	Height of output image the request refers to, in pixels	PositiveInteger type [c]	One (mandatory)
projection Projection	Camera specification and projection parameters	Projection type, see Table 8 and Table 9 Supported Projections are specified in service metadata	One (mandatory)
type Type	Type of navigation for computing new projection	ows:IdentifierType Supported navigation types are specified in service metadata	One (mandatory)
positions Positions	List of 2D positions for which to retrieve a new projection specification	List of CharacterString type, not empty List of list of coordinate values of length 2, in image CS, overall length is specified in server’s meatadat	One (mandatory) Include when requesting 2D position(s) desired
format Format	Format encoding of the result	ows:MimeType, see [OGC 06-121r3] clause 10.5 Default is “text/xml”	Zero or one (optional) Include when other than “text/xml” desired

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

NOTE 3 The UML class diagram contained in Subclause C.11 provides a useful graphical view of the contents of the GetCamera operation request listed in Table 62.

The “Multiplicity and use” columns in Table 62 specify the optionality of each listed parameter and data structure in the GetCamera operation request. All the “mandatory” parameters and data structures shall be implemented by all WVS clients, using a specified value(s). Similarly, all the “mandatory” parameters and data structures shall be implemented by all WVS servers, checking that each request parameter or data structure is received with any specified value(s).

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

14.2.2 GetCamera request KVP encoding (optional)

Servers may implement HTTP GET transfer of the GetCamera operation request, using KVP encoding. The KVP encoding of the GetCamera operation request shall use the parameters specified in Table 63. The parameters listed in Table 63 shall be as specified in Table 62 above.

Table 63 — GetCamera operation request URL parameters

Name and example [a]	Optionality and use	Definition and format
service=WCTS	Mandatory	Service type identifier
request= GetCamera	Mandatory	Operation name
version=0.6.0	Mandatory	Standard and schema version for this operation
GetViewRequestPart	Mandatory	Partial copy of the request parameters that generated the view for which information is desired
type=GoTo	Mandatory	Type of navigation for computing new projection
positions=211,423	Mandatory	2D position(s) in the image for which to retrieve 3D position(s) in the scene
format=text/plain	Optional	Format encoding of the result
a All parameter names are here listed using mostly lower case letters. However, any parameter name capitalization shall be allowed in KVP encoding, see Subclause 11.5.2 of [OGC 06-121r3].		

EXAMPLE An example GetCamera operation request KVP encoded for HTTP GET is:

```
http://hostname:port/path?SERVICE=WVS&VERSION=0.6.0&REQUEST=GetCamera&G
ETVIEWREQUESTPART&TYPE=GoTo&POSITIONS=27,12,35,89&FORMAT=text/plain
```

14.2.3 GetCamera request XML encoding (mandatory)

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

```
<!-- ============================================================= -->
<element name="GetCamera" type="wvs:GetCameraType"/>
<!-- ============================================================= -->
<complexType name="GetCameraType">
	<annotation>
		<documentation>XML encoded WVS GetFeatureInfo operation request. </documentation>
	</annotation>
	<complexContent>
		<extension base="wvs:WVSRequestBaseType">
			<sequence>
				<element ref="wvs:GetViewRequestPart"/>
				<element name="Type" type="ows:CodeType"/>
				<element name="PositionLists" type="wvs:Position2DListType" maxOccurs="unbounded"/>
				<element name="Format" type="ows:MimeType" minOccurs="0"/>
			</sequence>
		</extension>
	</complexContent>
</complexType>
```

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

```
<?xml version="1.0" encoding="UTF-8"?>
<GetCamera xmlns="http://www.opengis.net/wvs/0.6.0"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://www.opengis.net/wvs/0.6.0 ../wvsGetCamera.xsd"
	service="WVS" version="0.6.0" request="">
	<GetViewRequestPart />
	<Type>GoTo</Type>
	<PositionLists>
		<Position>27 12</Position>
		<Position>35 89/Position>
	</PositionLists>
	<Format>text/plain</Format>
</GetCamera>
```

14.3 GetCamera operation response

14.3.1 Normal response parameters

The normal response to a valid GetCamera operation request shall be a document containing a Projection specification formatted according to the Format parameter in the GetCamera request. More precisely, a response from the GetCamera operation shall include the parts listed in Table 64. This table also specifies the UML model data type plus the multiplicity and use of each listed part.

Table 64 — Parts of GetCamera operation response

Names	Definition	Data type and values	Multiplicity and use
projection Projection	Camera and projection specification	ProjectionBaseType E.g., PerspectiveProjection or OrthographicProjection, see Table 8 and Table 9	One (mandatory)

NOTE	The UML class diagram contained in Subclause C.11 provides a graphical view of the contents of the GetCamera operation response listed in Table 64.

14.3.2 Normal response XML encoding

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

```
<!-- ============================================================= -->
<element name="CameraResponse" type="wvs:CameraResponseType"/>
<!-- ============================================================= -->
<complexType name="CameraResponseType">
        <sequence>
                <element ref="wvs:ProjectionBase"/>
        </sequence>
</complexType>
```

14.3.3 GetCamera response example

A GetCamera operation response for WVS can look like this encoded in XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<CameraResponse xmlns="http://www.opengis.net/wvs/0.6.0"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.opengis.net/wvs/0.6.0 ../wvsGetCamera.xsd">
        <PerspectiveProjection>
                <POC>100 100 100</POC>
                <POI>101 101 101</POI>
                <Up>0 0 1</Up>
                <FOVX>80</FOVX>
                <FOVY>60</FOVY>
                <NearPlane>0.0001</NearPlane>
                <FarPlane>10000.0</FarPlane>
        </PerspectiveProjection>
</CameraResponse>
```

14.3.4 GetCamera exceptions

When a WVS server encounters an error while performing a GetCamera operation, it shall return an exception report message as specified in Clause 8 of [OGC 06-121r3]. The allowed standard exception codes shall include those listed in Table 65. For each listed exceptionCode, the contents of the "locator" parameter value shall be as specified in the right column of Table 65.

NOTE To reduce the need for readers to refer to other documents, the first four values listed below are copied from Table 25 in Subclause 8.3 of [OGC 06-121r3].

Table 65 — Exception codes for GetCamera operation

exceptionCode value	Meaning of code	"locator" value
OperationNotSupported	Request is for an operation that is not supported by this server	Name of operation not supported
MissingParameterValue	Operation request does not include a parameter value, and this server did not declare a default value for that parameter	Name of missing parameter
InvalidParameterValue	Operation request contains an invalid parameter value	Name of parameter with invalid value
OptionNotSupported	Request is for an option that is not supported by this server	Identifier of option not supported
TBD	TBD	TBD
NoApplicableCode	No other exceptionCode specified by this service and server applies to this exception	None, omit "locator" parameter

15 GetLayerInfo operation (optional, SLD-enabled WVS only)

15.1 General

The purpose of the optional GetLayerInfo request is to collect information on the available attribute names and the values in the attribute table of a specific layer. The attribute table is managed by the WVS server in a database table, as dbf file, or otherwise. The entries in the attribute table can be linked to the objects in the 3D view that can be retrieved using the GetView request by mapping ObjectIds to FeatureIds.

The GetLayerInfo operation is only supported for those Layers for which the attribute queryable=”1” (true) has been defined or inherited. A WVS server shall respond with a properly formatted service exception (XML) response (code = OperationNotSupported) if it receives a GetLayerInfo request but does not support it.

The GetLayerInfo request contains a mandatory Layer parameter for identifying the data set from which attribute information shall be received and an optional ColumnName parameter. If only the Layer parameter is used, then the response of the request shall contain only a list of all available attribute or column names in the format specified by the Format parameter. The received attribute names can then be used to receive additional information on the available values in the attribute table. If additionally to the Layer parameter also a ColumnName parameter is present, then the attribute table shall be queried for all available values that the features in the layer may have. The response to such a request contains a full list of unique values. In this list no duplicate values must occur.

15.2 GetLayerInfo operation request

15.2.1 GetLayerInfo request parameters

A request to perform the GetLayerInfo operation shall use the data structure specified in Table 66. This table also specifies the UML model data type, source of values, and multiplicity of each listed parameter, plus the meaning to servers when each optional parameter is not included in the operation request.

NOTE 1 To reduce the need for readers to refer to other documents, the first three parameters listed below are largely copied from Table 26 in Subclause 9.2.1 of [OGC 06-121r3].

Table 66 — Parameters in GetLayerInfo operation request

Names	Definition	Data type and values	Multiplicity and use
service service	Service type identifier	Character String type, not empty Value shall be "WVS"	One (mandatory)
request request	Operation name	Character String type, not empty Value shall be "GetLayerInfo"	One (mandatory)
version version	Standard version for operation	Character String type, not empty Value is specified by each Implementation Standard and Schemas version	One (mandatory)
layer Layer	Identifier of queryable Layer to get information from	Character String type, not empty Values are specified in server metadata	One (mandatory)
columNames ColumnNames	List of Column names to get information from	List of Character String type Column names. Default: empty	Zero or one (optional) Include when values for specific or all column names desired
format Format	Format encoding of the result	ows:MimeType, see [OGC 06-121r3] clause 10.5	One (mandatory)

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

NOTE 3 The UML class diagram contained in Subclause C.12 provides a useful graphical view of the contents of the GetLayerInfo operation request listed in Table 66.

The "Multiplicity and use" columns in Table 66 specify the optionality of each listed parameter and data structure in the GetLayerInfo operation request. All the "mandatory" parameters and data structures shall be implemented by all WVS clients, using a specified value(s). Similarly, all the "mandatory" parameters and data structures shall be implemented by all WVS servers, checking that each request parameter or data structure is received with any specified value(s).

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

15.2.1.1 Layer

The mandatory Layer parameter specified the layer from which information shall be retrieved. The parameter value shall refer to a layer identifier as described in the server's metadata; the Layer shall have the attribute queryable="1" (true) defined or inherited. If information from multiple layers needs to be collected, then multiple requests must be sent to the WVS server. If the layer identifier is not defined in the server's metadata, then server shall issue a service exception (code = UnknownLayer).

15.2.1.2 ColumnNames

The optional ColumnNames parameter specified one or several table column(s) or attribute name(s) of the selected layer in the Layer parameter from which all available unique values shall be retrieved. The value is a comma-separated list of one or more attribute names. If all available attributes of a layer shall be queried, the value of the ColumnNames parameter can be set to "ALLINFO". Default value is "ALLINFO". If any layer in the ColumnNames parameter is not defined in the service metadata of the server, the server shall issue a service exception (code = ColumnNameNotDefined).

COMMENT Maybe "columns" should be better renamed to "attributes", as a table containing columns is only one representation of an entity and its attributes.

15.2.1.3 Format

The mandatory Format parameter specifies the target encoding of the returned column and attribute information provided as ows:MimeType, see [OGC 06-121r3] clause 10.5. Available formats are described in the server's metadata.

15.2.2 GetLayerInfo request KVP encoding (optional)

Servers may implement HTTP GET transfer of the GetLayerInfo operation request, using KVP encoding. The KVP encoding of the GetLayerInfo operation request shall use the parameters specified in Table 67. The parameters listed in Table 67 shall be as specified in Table 66 above.

Table 67 — GetLayerInfo operation request URL parameters

Name and example [a]	Optionality and use	Definition and format
service=WVS	Mandatory	Service type identifier
request= GetLayerInfo	Mandatory	Operation name
version=0.6.0	Mandatory	Standard and schema version for this operation
layer=bldgs	Mandatory	The layer to get information from
columnnames=id,category	Optional	A list of column names to get information from
format=text/xml	Mandatory	Format encoding of the result
a All parameter names are here listed using mostly lower case letters. However, any parameter name capitalization shall be allowed in KVP encoding, see Subclause 11.5.2 of [OGC 06-121r3].		

EXAMPLE An example GetLayerInfo operation request KVP encoded for HTTP GET is:

```
http://hostname:port/path?SERVICE=WVS&VERSION=0.6.0&REQUEST=GetLayerInf
o&LAYER=bldgs&COLUMNNAMES=id,category&FORMAT=text/xml
```

15.2.3 GetLayerInfo request XML encoding (mandatory)

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

```
<!-- =========================================================== -->
<element name="GetLayerInfo" type="wvs:GetLayerInfoType"/>
<!-- =========================================================== -->
<complexType name="GetLayerInfoType">
	<annotation>
		<documentation>XML encoded W3DS GetLayerInfo operation request. </documentation>
	</annotation>
	<complexContent>
		<extension base="wvs:WVSRequestBaseType">
			<sequence>
				<element name="Layer" type="ows:CodeType"/>
				<element name="ColumnNames" type="wvs:IdentifierListType"/>
				<element name="Format" type="ows:MimeType"/>
			</sequence>
		</extension>
	</complexContent>
</complexType>
```

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

```
<?xml version="1.0" encoding="UTF-8"?>
<wvs:GetLayerInfo service="WVS" request="GetLayerInfo" version="0.6.0">
	<wvs:Layer>Buildings</wvs:Layer>
	<wvs:ColumnNames>
		<ows:Identifier>id</ows:Identifier>
		<ows:Identifier>category</ows:Identifier>
	</wvs:ColumnNames>
	<wvs:Format>text/xml</wvs:Format>
</wvs:GetLayerInfo>
```

15.3 GetLayerInfo operation response

15.3.1 Normal response parameters

The normal response to a valid GetLayerInfo operation request shall be a document encoded in MIME type text/xml containing attribute names and optionally attribute values of the selected layer. More precisely, a response from the GetLayerInfo operation shall include the parts listed in Table 68. This table also specifies the UML model data type plus the multiplicity and use of each listed part.

Table 68 — Parts of GetLayerInfo operation response

Names	Definition	Data type and values	Multiplicity and use
layerInfo LayerInfo	Base xml element	LayerInfo data type	One (mandatory)

Table 69 — Parts of LayerInfo data structure

Names	Definition	Data type and values	Multiplicity and use
li_layer LI_Layer	Layer from which attribute data was collected	LI_Layer data type	One (mandatory)

Table 70 — Parts of LI_Layer data structure

Names	Definition	Data type and values	Multiplicity and use
identifier Identifier	Identifier of this layer	Character String type, not empty	One (mandatory)
attribute Attribute	Attribute or column found in the server's data repository for this layer	Attribute data type	Zero or more (optional)

Table 71 — Parts of Attribute data structure

Names	Definition	Data type and values	Multiplicity and use
name Name	Name of this attribute	Character String type, not empty	One (mandatory)
type Type	Data type of this attribute	Character String type, not empty	One (mandatory)
uniqueCount UniqueCount	Number of unique values for this attribute	Long type	Zero or one (optional)
values Values	List of unique values stored in the server's data repository for the selected layer and attribute/column	Values data type	Zero or one (optional)

Table 72 — Parts of Values data structure

Names	Definition	Data type and values	Multiplicity and use
value Value	Unique attribute value	Character String type, may be empty	Zero or more (optional)

NOTE The UML class diagram contained in Subclause C.12 provides a graphical view of the contents of the GetLayerInfo operation response listed in Table 68 to Table 72.

15.3.2 Normal response XML encoding

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

```
<!-- ========================================================= -->
<element name="LayerInfo" type="wvs:LayerInfoType"/>
<!-- ========================================================= -->
<complexType name="LayerInfoType">
	<sequence>
		<element name="LI_Layer" type="wvs:LI_Layer" minOccurs="0" maxOccurs="unbounded"/>
	</sequence>
</complexType>
<!-- ========================================================= -->
```

```
<complexType name="LI_Layer">
	<sequence>
		<element ref="ows:Identifier"/>
		<element name="Attribute" type="wvs:Attribute" minOccurs="0" maxOccurs="unbounded"/>
	</sequence>
</complexType>
<!-- ============================================================== -->
<complexType name="Attribute">
	<sequence>
		<element name="Name" type="ows:CodeType"/>
		<element name="Type" type="string"/>
		<element name="Values" type="wvs:Values" minOccurs="0" maxOccurs="1"/>
		<element name="UniqueCount" type="long" minOccurs="0" maxOccurs="1"/>
	</sequence>
</complexType>
<!-- ============================================================== -->
<complexType name="Values">
	<sequence>
		<element name="Value" type="string" minOccurs="0" maxOccurs="unbounded"/>
	</sequence>
</complexType>
```

15.3.3 GetLayerInfo response example

A GetLayerInfo operation response for VWS can look like this encoded in XML:

EXAMPLE 1 An example response to a GetLayerInfo request, including a Layer parameter only, XML encoded for HTTP POST is:

```
<?xml version="1.0" encoding="UTF-8"?>
<wvs:LayerInfo>
	<wvs:LI_Layer>
		<ows:Identifier>dem</ows:Identifier>
		<wvs:Attribute>
			<wvs:Name>id</wvs:Name>
			<wvs:Type>Long</wvs:Type>
			<wvs:UniqueCount>129764</wvs:UniqueCount>
		</wvs:Attribute>
		<wvs:Attribute>
			<wvs:Name>landuse</wvs:Name>
			<wvs:Type>String</wvs:Type>
			<wvs:UniqueCount>5</wvs:UniqueCount>
		</wvs:Attribute>
	</wvs:LI_Layer>
</wvs:LayerInfo>
```

EXAMPLE 2 An example response to a GetLayerInfo request, including a Layer parameter only, XML encoded for HTTP POST is:

```
<?xml version="1.0" encoding="UTF-8"?>
<wvs:LayerInfo>
	<wvs:LI_Layer>
		<ows:Identifier>dem</ows:Identifier>
		<wvs:Attribute>
			<wvs:Name>landuse</wvs:Name>
			<wvs:Type>String</wvs:Type>
			<wvs:Values>
				<wvs:Value>train</wvs:Value>
				<wvs:Value>block</wvs:Value>
				<wvs:Value>green</wvs:Value>
				<wvs:Value>street</wvs:Value>
				<wvs:Value>forest</wvs:Value>
			</wvs:Values>
			<wvs:UniqueCount>129764</wvs:UniqueCount>
		</wvs:Attribute>
	</wvs:LI_Layer>
</wvs:LayerInfo>
```

15.3.4 GetLayerInfo exceptions

When a WVS server encounters an error while performing a GetLayerInfo operation, it shall return an exception report message as specified in Clause 8 of [OGC 06-121r3]. The allowed standard exception codes shall include those listed in Table 73. For each listed exceptionCode, the contents of the "locator" parameter value shall be as specified in the right column of Table 73.

NOTE To reduce the need for readers to refer to other documents, the first four values listed below are copied from Table 25 in Subclause 8.3 of [OGC 06-121r3].

Table 73 — Exception codes for GetLayerInfo operation

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

16 GetLegendGraphic operation (optional, SLD-enabled WVS only)

16.1 General

The GetLegendGraphic operation allows a client to retrieve a graphic containing a map legend for an identified dataset and style. Implementation of this operation is optional by WVS servers. This GetLegendGraphic operation extends and renames the GetResource operation specified in Subclause 9.4 of the draft extended OWS Common Specification [OGC 06-121r3].

COMMENT The GetLegendGraphic is mainly copied from draft WPVS specification [7].

16.2 GetLegendGraphic operation request

16.2.1 GetLegendGraphic request parameters

A request to perform the GetLegendGraphic operation shall include the data structure specified in Table 74. This table also specifies the UML model data type, source of values, and multiplicity of each listed parameter, plus the meaning to servers when each optional parameter is not included in the operation request. Although some values listed in the “Names” column appear to contain spaces, they shall not contain spaces.

NOTE 1 To reduce the need for readers to refer to other documents, the first three parameters listed below are largely copied from Table 26 in Subclause 9.2.1 of [OGC 06-121r3].

Table 74 — Parameters in GetLegendGraphic operation request

Names [a]	Definition	Data type and values	Multiplicity and use
service service	Service type identifier	Character String type, not empty Value shall be "WVS"	One (mandatory)
request request	Operation name	Character String type, not empty Value shall be "GetLegendGraphic"	One (mandatory)
version version	Standard version for operation	Character String type, not empty Value is specified by each Implementation Standard and Schemas version	One (mandatory)
layer Layer	Identifier of data Layer for which to produce legend graphic	ows:CodeType Values defined in service metadata or in other metadata known to client	One (mandatory)
style Style	Identifier of style for which to produce legend graphic	Character String type, not empty Values defined in service metadata (Capabilities) or in SLD file	Zero or one (optional) Include if not default style for this layer
featureType FeatureType	Identifier of feature type for which to produce legend graphic	Character String type, not empty	Zero or one (optional) Include if more than one feature type
rule Rule	Identifier of rule of style for which to produce legend graphic	Character String type, not empty Values defined in SLD file	Zero or one (optional) Include if specific rule desired
scaleDenominator Scale Denominator	Approximate map scale denominator of client dataset display	Double type, see [OGC 05-077r4] Values defined in Layers metadata	Zero or one (optional) Include if Rule not specified and scale is useful
sld SLD	Styled layer descriptor file	URL or file contents	Zero or one (optional)
sld_body SLD_Body	SLD document in HTTP GET request	<StyledLayerDescriptor> TBD: As in "SLD profile for WMS" [OGC 05-078r4] spec?	Zero or one (optional) Include when SLD document in HTTP GET request desired
format Format	Reference to format in which legend graphic should be encoded	MIME type, see [OGC 06-121r3] clause 10.5 Values are specified in service metadata	One (mandatory)
width Width	Width of desired legend graphic, in pixel	PositiveInteger type [b]	Zero or one (optional) Include when other than default width desired

height Height	Height of desired legend graphic, in pixels	PositiveInteger type [b]	Zero or one (optional) Include when other than default height desired
exceptions Exceptions	Reference to format in which operation exceptions should be returned	CodeList type, either “XML” “INIMAGE” “BLANK” Default is XML	Zero or one (optional) Include when other than XML desired
a The name capitalization rules being used here are specified in Subclause 11.6.2 of [OGC 05-008r1]. b Value ranges for Width and Height may be specified using the Parameter element in the Operation element in the OperationsMetadata section of the service metadata document.			

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

NOTE 3 The UML class diagram contained in Subclause C.13 provides a useful graphical view of the contents of the GetLegendGraphic operation request listed in Table 74.

The “Multiplicity and use” columns in Table 74 specify the optionality of each listed parameter and data structure in the GetLegendGraphic operation request. All the “mandatory” parameters and data structures shall be implemented by all WVS clients, using a specified value(s). Similarly, all the “mandatory” parameters and data structures shall be implemented by all WVS servers, checking that each request parameter or data structure is received with any specified value(s).

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

16.2.1.1 Exceptions

The operations parameter Exceptions parameter states the format in which to report errors during processing a GetLegendGraphic request as one of “XML” (default), “INIMAGE”, or “BLANK” (see clause 7.2.5).

16.2.2 GetLegendGraphic request KVP encoding (optional)

Servers may implement HTTP GET transfer of the GetLegendGraphic operation request, using KVP encoding. The KVP encoding of the GetLegendGraphic operation request shall use the parameters specified in Table 75. The parameters listed in Table 75 shall be as specified in Table 74 above.

Table 75 — GetLegendGraphic operation request URL parameters

Name and example [a]	Optionality and use	Definition and format
service=WVS	Mandatory	Service type identifier

request= GetLegendGraphic	Mandatory	Operation name
version=0.6.0	Mandatory	Standard and schema version for this operation
layer=Bldgs	Mandatory	Identifier of Layer for which to produce legend graphic
style=usage	Optional, include if not default style for this Layer	Identifier of Style for which to produce legend graphic
namespace=xmlns(bldg= http://www.opengis.net/citygml/building/1.0)	Mandatory if FeatureType parameter used	Identifier of namespace of FeatureType parameter
featuretype=bldg:Building	Optional, include if more than one feature type in that Layer	Identifier of feature type for which to produce legend graphic
rule=colorcoded	Optional, include if not	Identifier of Rule for which to produce legend graphic
scaledenominator=5000	Optional, include if Rule not specified and scale is useful	Approximate map scale denominator of client Layer display
sld=http://myhost/ myBldgStyle.xml	Optional, mutally exclusive with SLD_Body [b]	URL reference to SLD document
sld_body=TBD	Optional, mutally exclusive with SLD [b]	
format=image/png	Mandatory	MIME type of format in which legend graphic should be encoded
width=100	Optional, include when specific width desired	Width of legend graphic, in pixels
height=100	Optional, include when specific height desired	Height of legend graphic, in pixels
exceptions=INIMAGE	Optional, include when other than default XML desired	Reference to format in which operation exceptions should be returned
b Either SLD or SLD_Body shall be used		

EXAMPLE An example GetLegendGraphic operation request KVP encoded (not yet URL encoded) for HTTP GET is:

```
http://hostname:port/path?SERVICE=WVS&VERSION=0.6.0&REQUEST=GetLegendGr
aphic&LAYER=Buildings&STYLE=usage&NAMESPACE=xmlns(bldg=http://www.openg
is.net/citygml/building/1.0)FEATURETYPE=bldg:Building&RULE=colorcoded&S
LD=http://myhost/myBldgStyle.xml&FORMAT=image/png&/WIDTH=100&HEIGHT=100
&EXCEPTIONS=INIMAGE
```

16.2.3 GetLegendGraphic request XML encoding (mandatory)

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

```
<complexType name="GetLegendGraphicType">
	<annotation>
		<documentation>XML encoded WVS GetLegendGraphic operation request. </documentation>
	</annotation>
```

```
        <complexContent>
                <extension base="wvs:WVSRequestBaseType">
                        <sequence>
                                <element name="Layer" type="ows:CodeType"/>
                                <element name="Style" type="ows:CodeType" minOccurs="0"/>
                                <element name="FeatureType" type="ows:CodeType" minOccurs="0"/>
                                <element name="Rule" type="string" minOccurs="0"/>
                                <element name="ScaleDenominator" type="double" minOccurs="0"/>
                                <element ref="sld:StyledLayerDescriptor" minOccurs="0"/>
                                <element name="Format" type="ows:MimeType"/>
                                <element name="Width" type="integer" minOccurs="0"/>
                                <element name="Height" type="integer" minOccurs="0"/>
                                <element name="Excpetions" type="string" minOccurs="0"/>
                        </sequence>
                </extension>
        </complexContent>
</complexType>
```

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

```
<?xml version="1.0" encoding="UTF-8"?>
<GetLegendGraphic xmlns="http://www.opengis.net/wvs/0.6.0"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.opengis.net/wvs/0.6.0 ../wvsGetLegendGraphic.xsd"
        service="WVS" version="0.6.0" request="">
        <Layer>Bldgs</Layer>
        <Style>usage</Style>
        <FeatureType xmlns:bldg="http://www.opengis.net/citygml/building/1.0">bldg:Building</FeatureType>
        <ScaleDenominator>5000.0</ScaleDenominator>
        <SLD>http://myhost/myBldgStyle.xml</SLD>
        <Format>image/png</Format>
        <Width>100</Width>
        <Height>100</Height>
        <Excpetions>INIMAGE</Excpetions>
</GetLegendGraphic>
```

16.3 GetLegendGraphic operation response

16.3.1 Normal response parameters

The normal response to a valid GetLegendGraphic operation request shall be the desired legend graphic encoded in the desired image format or graphic format.

16.3.2 GetLegendGraphic exceptions

When a WVS server encounters an error while performing a GetLegendGraphic operation, it shall return an exception report message as specified in Clause 8 of [OGC 06-121r3]. The allowed standard exception codes shall include those listed in Table 76. For each listed exceptionCode, the contents of the "locator" parameter value shall be as specified in the right column of Table 76.

NOTE To reduce the need for readers to refer to other documents, the first four values listed below are copied from Table 25 in Subclause 8.3 of [OGC 06-121r3].

Table 76 — Exception codes for GetLegendGraphic operation

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

MissingParameterValue	Operation request does not include a parameter value, and this server did not declare a default value for that parameter	Name of missing parameter
InvalidParameterValue	Operation request contains an invalid parameter value	Name of parameter with invalid value
OptionNotSupported	Request is for an option that is not supported by this server	Identifier of option not supported
TBD	TBD	TBD
NoApplicableCode	No other exceptionCode specified by this service and server applies to this exception	None, omit "locator" parameter

Annex A
(normative)

Abstract test suite

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

Inclusion of the Abstract Test Suite is expected in version 1.0.0 of each OGC Implementation Standard. In earlier versions, the following paragraph can be used:

An abstract test suite is not provided in this version of this Implementation Standard, but will be provided in version 1.0.0.

Annex B
(normative)

XML Schema Documents

In addition to this document, this standard includes several normative XML Schema Documents. These XML Schema Documents are bundled in a zip file with the present document. After OGC acceptance of a Version 1.0.0 of this standard, these XML Schema Documents will also be posted online at the URL:

http://schemas.opengis.net/WVS/0.6.0.

In the event of a discrepancy between the bundled and online versions of the XML Schema Documents, the online files shall be considered authoritative.

The WVS abilities now specified in this document use nine specified XML Schema Documents included in the zip file with this document. These XML Schema Documents combine the XML schema fragments listed in various subclauses of this document, eliminating duplications. These XML Schema Documents roughly match the 10 UML packages described in Annex B, and are named:

wvsCommon.xsd

wvsGetCapabilities.xsd

wvsGetView.xsd

wvsGetFeatureInfo.xsd

wvsGetPosition

wvsGetMeasurement

wvsGetCamera

wvsGetLayerInfo.xsd

wvsGetLegendGraphic.xsd

These XML Schema Documents use and build on the OWS common XML Schema Documents specified [OGC 06-121r3], 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 06-121r3].

Annex C
(informative)

UML model

TBD: UML models have to be reviewed and formatted according to the requirements.

C.1 Introduction

This annex provides a UML model of the WVS interface, using the OGC/ISO profile of UML summarized in Subclause 5.2 of [06-121r3].

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

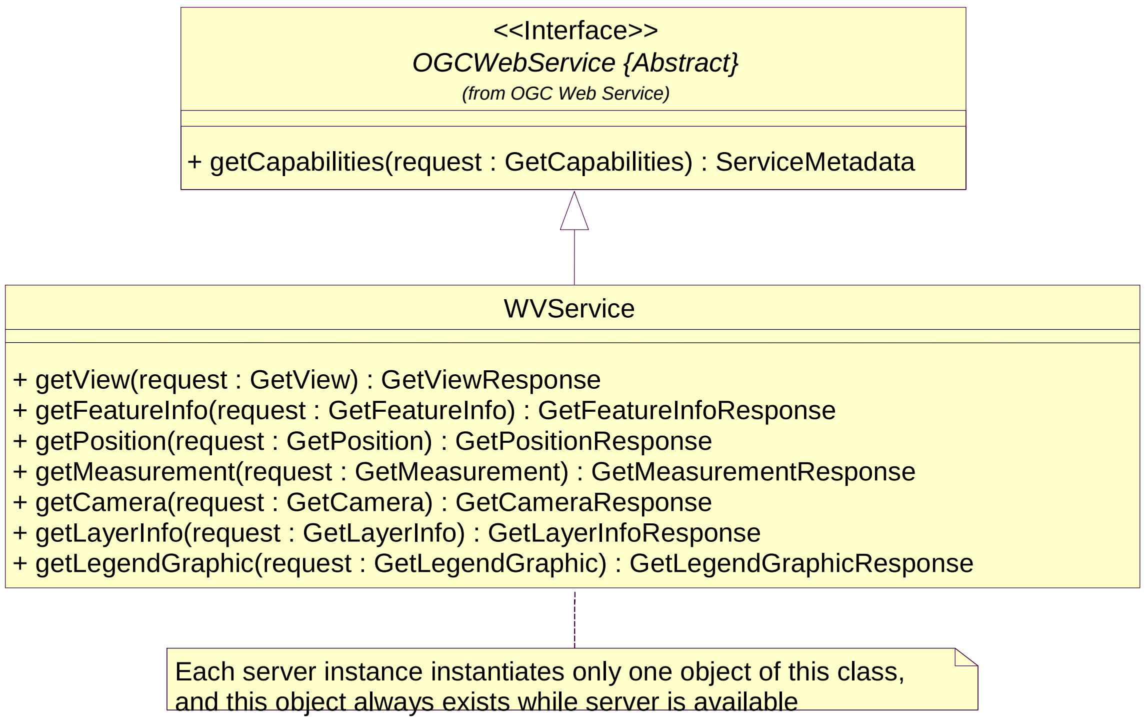

<<Interface>>
OGCWebService {Abstract}
(from OGC Web Service)
+ getCapabilities(request : GetCapabilities) : ServiceMetadata
WVService
+ getView(request : GetView) : GetViewResponse
+ getFeatureInfo(request : GetFeatureInfo) : GetFeatureInfoResponse
+ getPosition(request : GetPosition) : GetPositionResponse
+ getMeasurement(request : GetMeasurement) : GetMeasurementResponse
+ getCamera(request : GetCamera) : GetCameraResponse
+ getLayerInfo(request : GetLayerInfo) : GetLayerInfoResponse
+ getLegendGraphic(request : GetLegendGraphic) : GetLegendGraphicResponse
Each server instance instantiates only one object of this class, and this object always exists while server is available

Figure C.1 — WVS interface UML diagram

Each of the WVS operations uses a request and a response data type, each of which is defined by one or more additional UML classes. The following subclauses provide a more complete UML model of the WVS interface, adding UML classes defining the operation request and response data types.

C.2 UML packages

The WVS interface UML model is organized in 10 packages, as shown in the package diagram in Figure C.2. These WVS-specific packages make use of five non-WVS-specific packages, named OWS Web Service, OWS Operations Metadata, OWS Service Identification, OWS Service Provider, and ISO 19115 Subset. This package diagram shows the dependencies among the various packages shown.

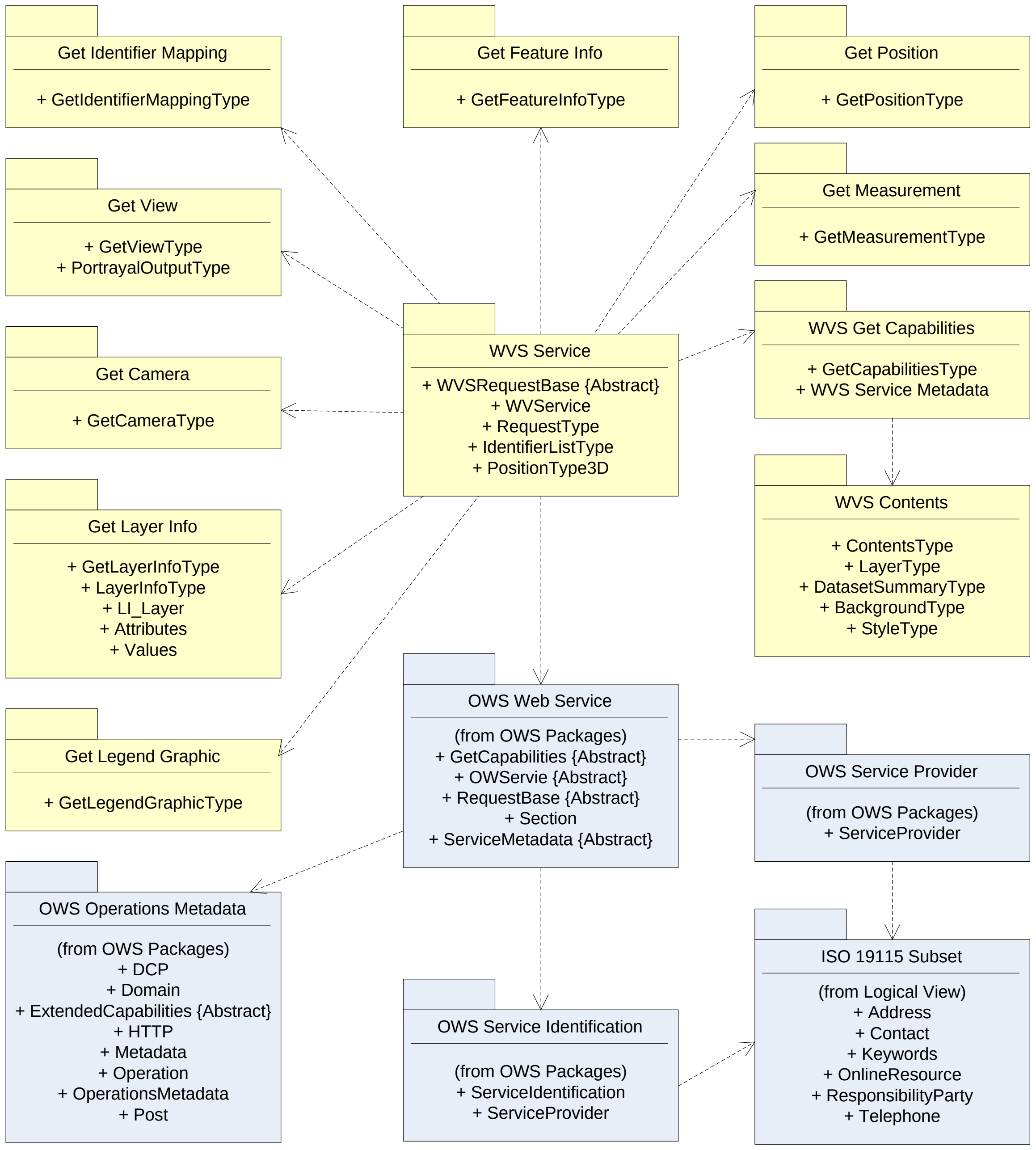


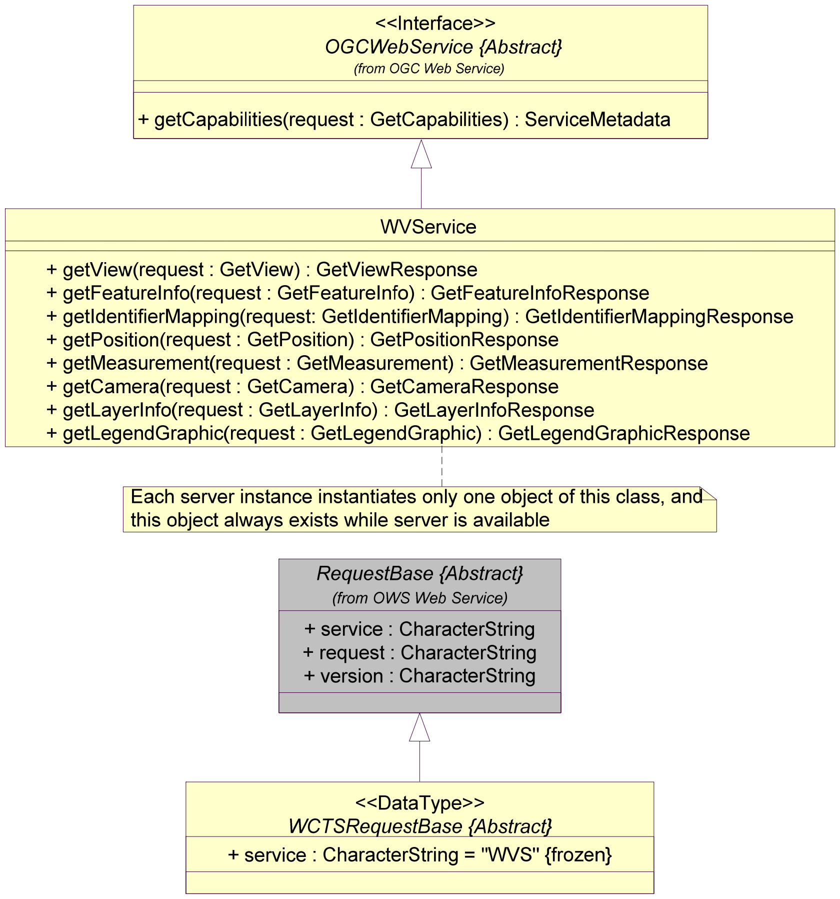
Figure C.2 — WVS interface package diagram

Each of the 10 WVS-specific packages shown in Figure C.2 is described in the following subclauses. The OWS Web Service, OWS Operations Metadata, OWS Service

Identification, OWS Service Provider, and ISO 19115 Subset packages are described in Annex C of [OGC 06-121r3].

C.3 WVS Service package

The WVS Service package is shown in the class diagram in Figure C.3. This diagram does not show the classes used by the WVS operation requests and responses, which are shown (with part of this package) in the GetView, GetFeatureInfo, GetPosition, GetMeasurement, GetCamera, GetLayerInfo, and GetLegendGraphic packages. This diagram also shows one used class from the OWS Web Service package, which is common to all OGC Web Services, plus one used class from the WVS package.

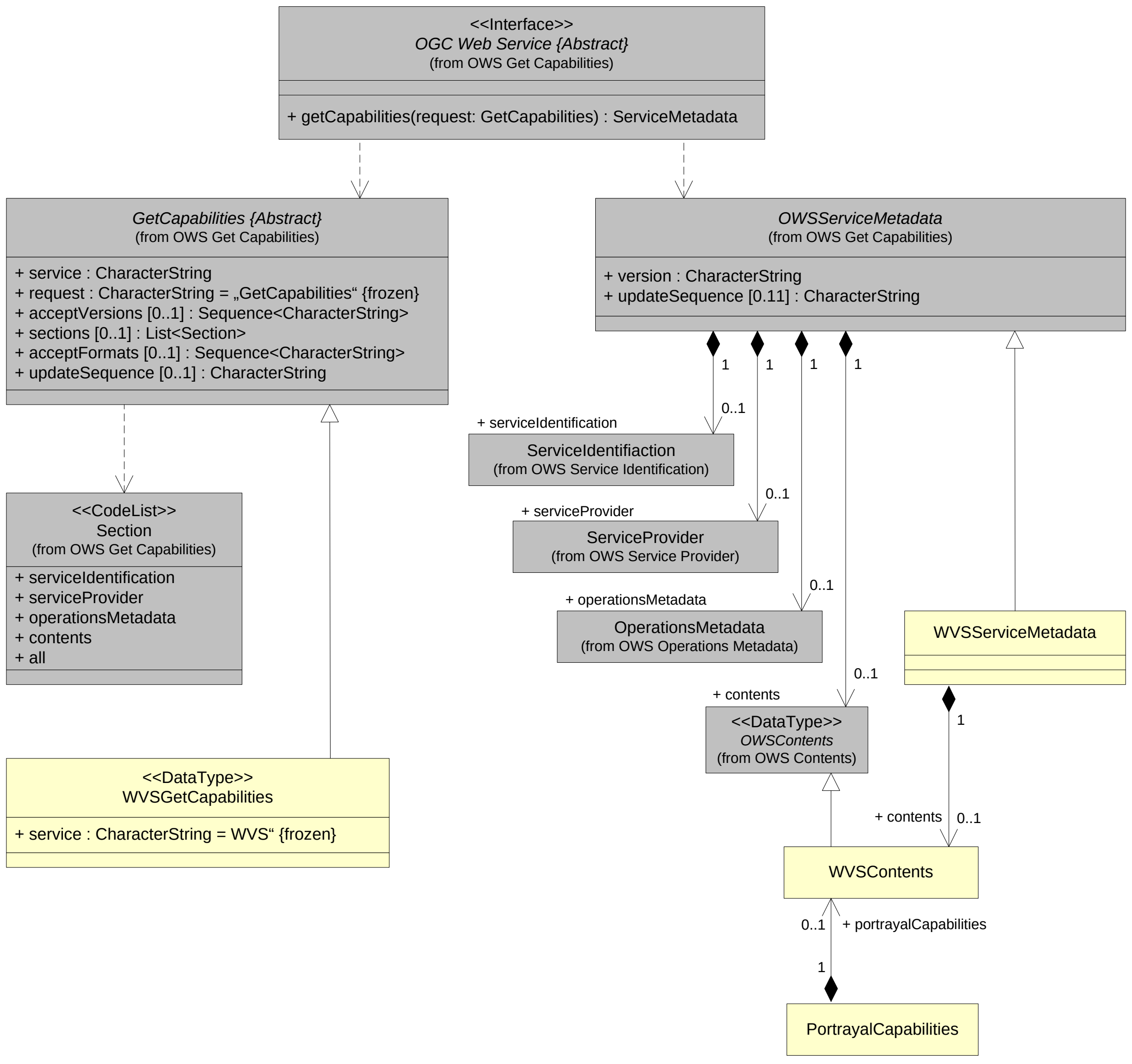


<<Interface>>
OGCWebService {Abstract}
(from OGC Web Service)
+ getCapabilities(request : GetCapabilities) : ServiceMetadata
WVService
+ getView(request : GetView) : GetViewResponse
+ getFeatureInfo(request : GetFeatureInfo) : GetFeatureInfoResponse
+ getIdentifierMapping(request: GetIdentifierMapping) : GetIdentifierMappingResponse
+ getPosition(request : GetPosition) : GetPositionResponse
+ getMeasurement(request : GetMeasurement) : GetMeasurementResponse
+ getCamera(request : GetCamera) : GetCameraResponse
+ getLayerInfo(request : GetLayerInfo) : GetLayerInfoResponse
+ getLegendGraphic(request : GetLegendGraphic) : GetLegendGraphicResponse
Each server instance instantiates only one object of this class, and this object always exists while server is available
RequestBase {Abstract}
(from OWS Web Service)
+ service : CharacterString
+ request : CharacterString
+ version : CharacterString
<<DataType>>
WCTSRequestBase {Abstract}
+ service : CharacterString = "WVS" {frozen}

Figure C.3 — WVS Service package class diagram

C.4 WVS Get Capabilities package

The WVS Get Capabilities package is shown in the class diagram in Figure C.4. This diagram also shows several classes from the OWS Get Capabilities package. The WVSGetCapabilities class introduced by this package is further defined by Table 14 in this document.



<<Interface>>
OGC Web Service {Abstract}
(from OWS Get Capabilities)
+ getCapabilities(request: GetCapabilities) : ServiceMetadata
GetCapabilities {Abstract}
(from OWS Get Capabilities)
+ service : CharacterString
+ request : CharacterString = „GetCapabilities“ {frozen}
+ acceptVersions [0..1] : Sequence<CharacterString>
+ sections [0..1] : List<Section>
+ acceptFormats [0..1] : Sequence<CharacterString>
+ updateSequence [0..1] : CharacterString
OWSServiceMetadata
(from OWS Get Capabilities)
+ version : CharacterString
+ updateSequence [0.11] : CharacterString
1
0..1
+ serviceIdentification
ServiceIdentifiaction
(from OWS Service Identification)
+ serviceProvider
ServiceProvider
(from OWS Service Provider)
+ operationsMetadata
OperationsMetadata
(from OWS Operations Metadata)
+ contents
<<DataType>>
OWSContents
(from OWS Contents)
<<CodeList>>
Section
(from OWS Get Capabilities)
+ serviceIdentification
+ serviceProvider
+ operationsMetadata
+ contents
+ all
<<DataType>>
WVSGetCapabilities
+ service : CharacterString = WVS“ {frozen}
WVSServiceMetadata
WVSContents
+ portrayalCapabilities
PortrayalCapabilities

Figure C.4 — WVS Get Capabilities package class diagram

C.5 WVS Contents package

The WVS Contents package is shown in the class diagram in Figure C.5. This diagram also shows classes from the OWS Common and OWS DataIdentification package. The

Contents class introduced by this package is further defined by Table 18 through Table 23 in this document.

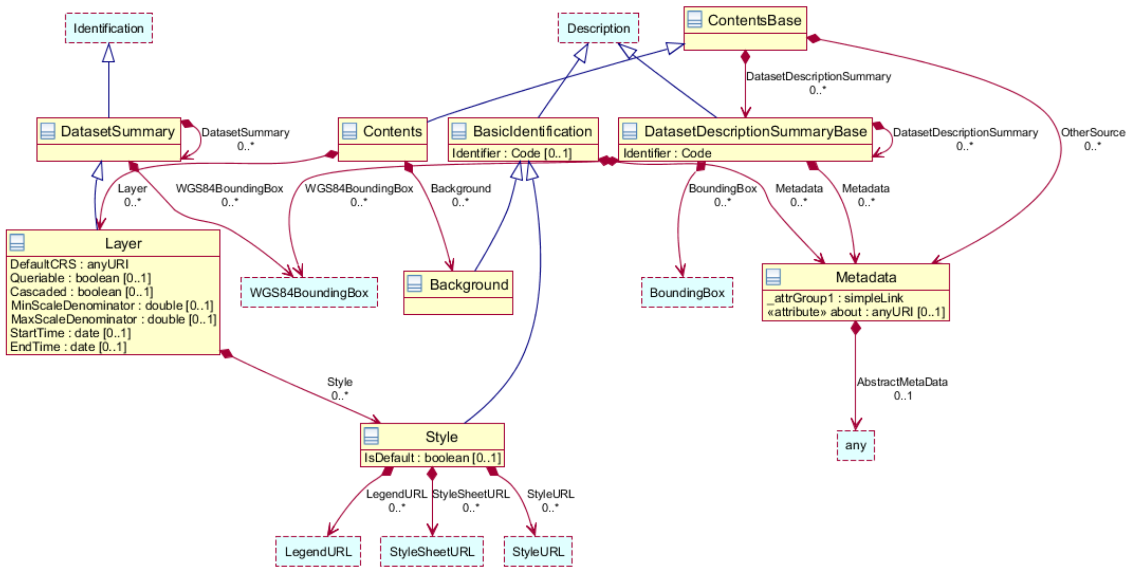


Figure C.5 — WVS Contents package class diagram

C.6 WVS Get View package

The WVS Get View package is shown in the class diagram in Figure C.6. This diagram also shows classes from the OWS Common and OWS Get Capabilities package. The GetView class introduced by this package is further defined by Table 31 in this document.

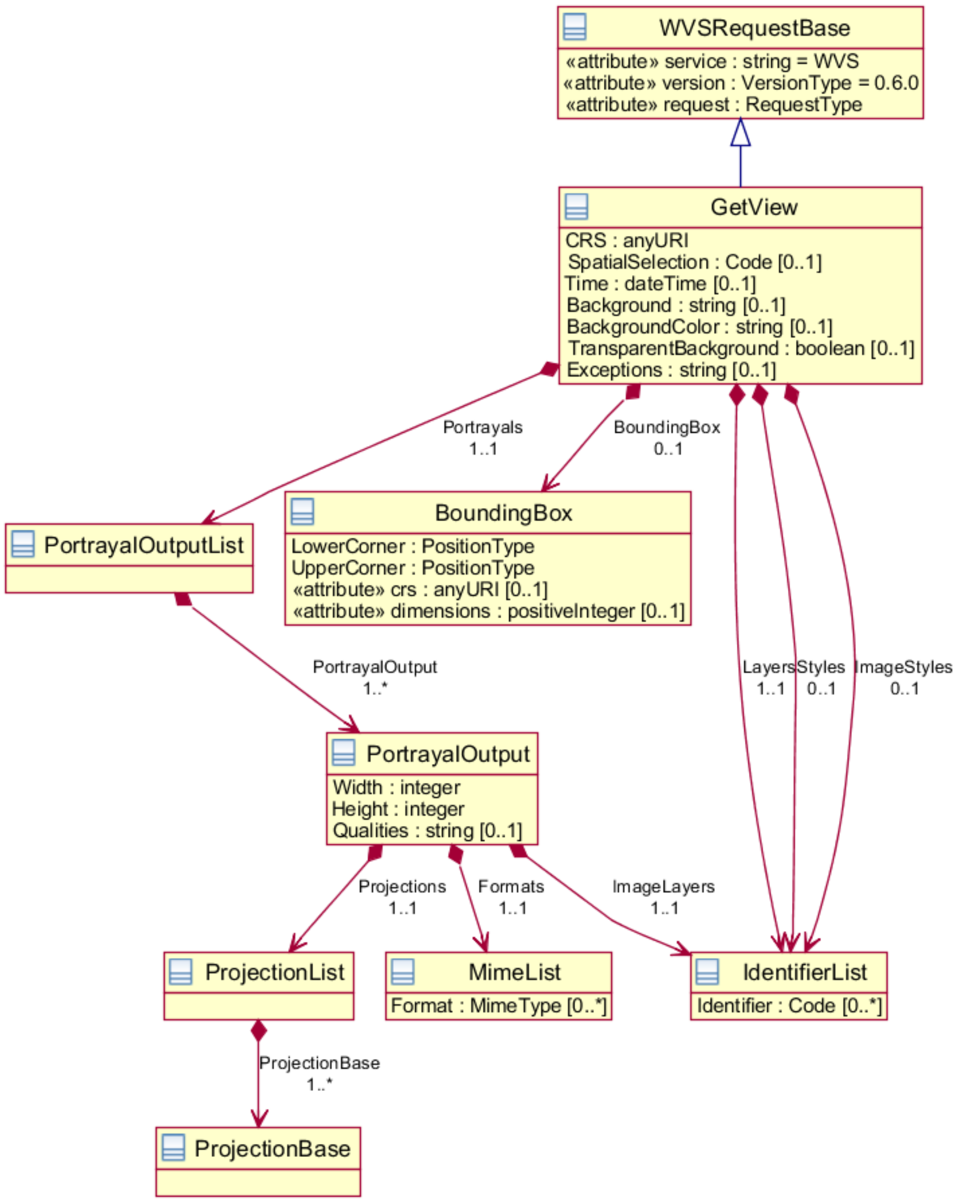

WVSRequestBase
«attribute» service : string = WVS
«attribute» version : VersionType = 0.6.0
«attribute» request : RequestType
GetView
CRS : anyURI
SpatialSelection : Code [0..1]
Time : dateTime [0..1]
Background : string [0..1]
BackgroundColor : string [0..1]
TransparentBackground : boolean [0..1]
Exceptions : string [0..1]
Portrayals
1..1
BoundingBox
0..1
PortrayalOutputList
BoundingBox
LowerCorner : PositionType
UpperCorner : PositionType
«attribute» crs : anyURI [0..1]
«attribute» dimensions : positiveInteger [0..1]
PortrayalOutput
1..*
LayersStyles
1..1
0..1
ImageStyles
0..1
PortrayalOutput
Width : integer
Height : integer
Qualities : string [0..1]
Projections
1..1
Formats
1..1
ImageLayers
1..1
ProjectionList
MimeList
Format : MimeType [0..*]
IdentifierList
Identifier : Code [0..*]
ProjectionBase
1..*
ProjectionBase

Figure C.6 — Get View package class diagram

C.7 WVS Get Feature Info package

The WVS Get Feature Info package is shown in the class diagram in Figure C.7. This diagram also shows classes from the OWS Common and OWS Get Capabilities package.

The GetFeatureInfo class introduced by this package is further defined by Table 36 in this document.

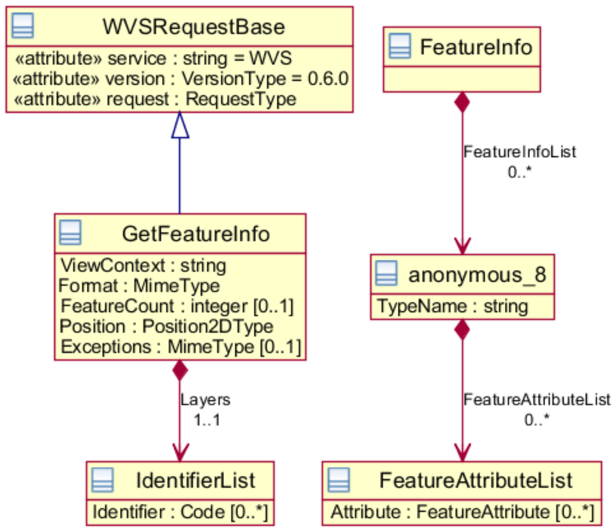

WVSRequestBase
«attribute» service : string = WVS
«attribute» version : VersionType = 0.6.0
«attribute» request : RequestType
FeatureInfo
FeatureInfoList
0..*
GetFeatureInfo
ViewContext : string
Format : MimeType
FeatureCount : integer [0..1]
Position : Position2DType
Exceptions : MimeType [0..1]
anonymous_8
TypeName : string
Layers
1..1
FeatureAttributeList
0..*
IdentifierList
Identifier : Code [0..*]
FeatureAttributeList
Attribute : FeatureAttribute [0..*]

Figure C.7 — WVS Get Feature Info package class diagram

C.8 WVS Get Identifier Mapping package

The WVS Get Identifier Mapping package is shown in the class diagram in Figure C.8. This diagram also shows classes from the OWS Common and OWS Get Capabilities package. The GetPosition class introduced by this package is further defined by Table 44 in this document.

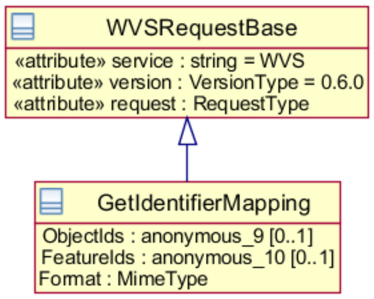

WVSRequestBase
«attribute» service : string = WVS
«attribute» version : VersionType = 0.6.0
«attribute» request : RequestType
GetIdentifierMapping
ObjectIds : anonymous_9 [0..1]
FeatureIds : anonymous_10 [0..1]
Format : MimeType

Figure C.8 — WVS Get Identifier Mapping package class diagram

C.9 WVS Get Position package

The WVS Get Position package is shown in the class diagram in Figure C.9. This diagram also shows classes from the OWS Common and OWS Get Capabilities package. The GetPosition class introduced by this package is further defined by Table 50 in this document.

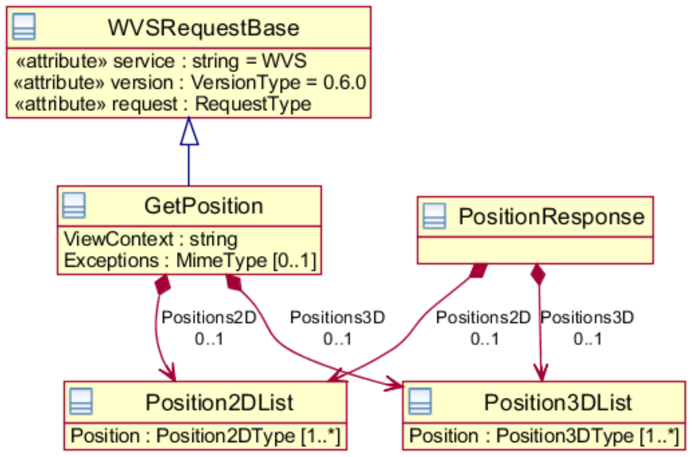

WVSRequestBase
«attribute» service : string = WVS
«attribute» version : VersionType = 0.6.0
«attribute» request : RequestType
GetPosition
ViewContext : string
Exceptions : MimeType [0..1]
PositionResponse
Positions2D
0..1
Positions3D
0..1
Positions2D
0..1
Positions3D
0..1
Position2DList
Position : Position2DType [1..*]
Position3DList
Position : Position3DType [1..*]

Figure C.9 — WVS Get Position package class diagram

C.10 WVS Get Measurement package

The WVS Get Measurement package is shown in the class diagram in Figure C.10. This diagram also shows classes from the OWS Common and OWS Get Capabilities package. The GetMeasurement class introduced by this package is further defined by Table 56 in this document.

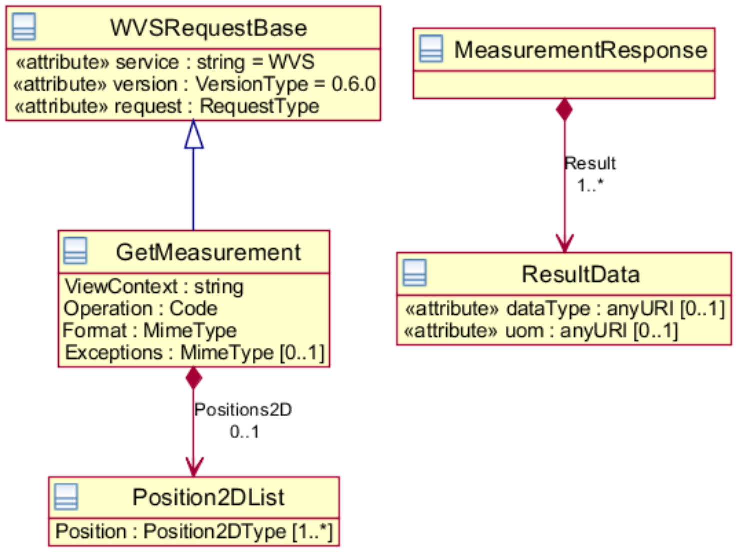

WVSRequestBase
«attribute» service : string = WVS
«attribute» version : VersionType = 0.6.0
«attribute» request : RequestType
MeasurementResponse
Result
1..*
GetMeasurement
ViewContext : string
Operation : Code
Format : MimeType
Exceptions : MimeType [0..1]
ResultData
«attribute» dataType : anyURI [0..1]
«attribute» uom : anyURI [0..1]
Positions2D
0..1
Position2DList
Position : Position2DType [1..*]

Figure C.10 — WVS Get Measurement package class diagram

C.11 WVS Get Camera package

The WVS Get Camera package is shown in the class diagram in Figure C.11. This diagram also shows classes from the OWS Common and OWS Get Capabilities package. The GetCamera class introduced by this package is further defined by Table 62 in this document.

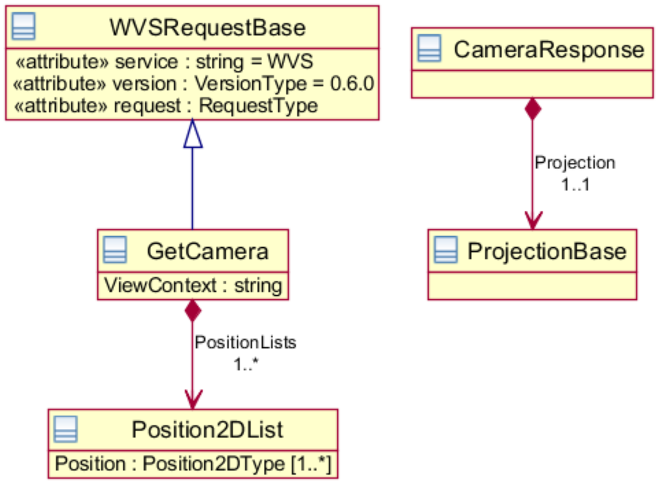

WVSRequestBase
«attribute» service : string = WVS
«attribute» version : VersionType = 0.6.0
«attribute» request : RequestType
CameraResponse
Projection
1..1
GetCamera
ViewContext : string
ProjectionBase
PositionLists
1..*
Position2DList
Position : Position2DType [1..*]

Figure C.11 — WVS Get Camera package class diagram

C.12 WVS Get Layer Info package

The WVS Get Layer Info package is shown in the class diagram in Figure C.12. This diagram also shows classes from the OWS Common and OWS Get Capabilities package. The GetLayerInfo class introduced by this package is further defined by Table 66 in this document.

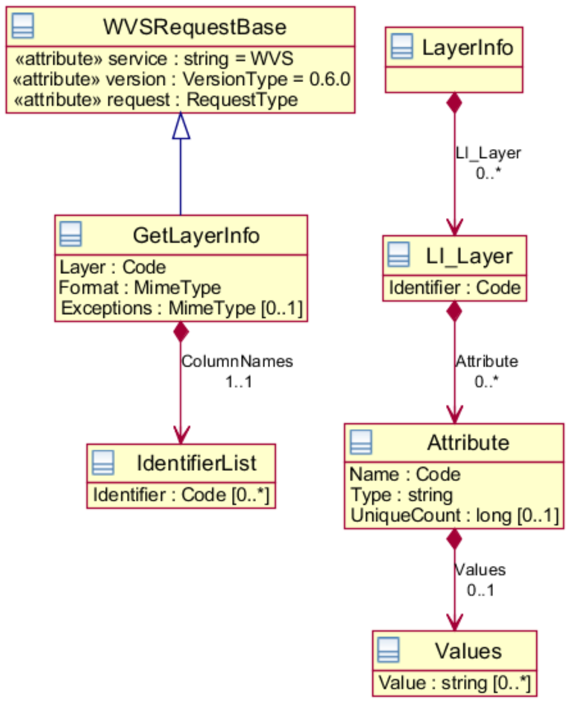


Figure C.12 — WVS Get Layer Info package class diagram

C.13 WVS Get Legend Graphic package

The WVS Get Legend Graphic package is shown in the class diagram in Figure C.4. This diagram also shows classes from the OWS Common and OWS Get Capabilities package. The GetLegendGraphic class introduced by this package is further defined by Table 74 in this document.

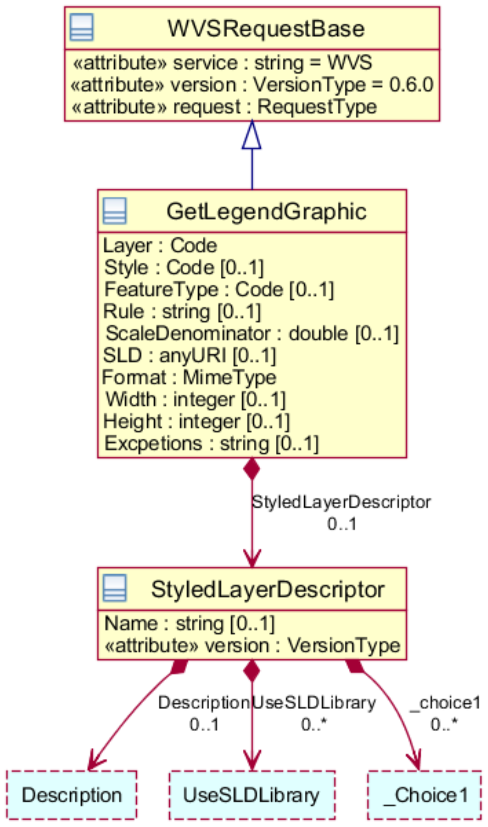


Figure C.13 — WVS Get Legend Graphic package class diagram

Annex D
(informative)

Example XML documents

D.1 Introduction

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

D.2 GetCapabilities response example

Example of a GetCapabilities response focusing on the PortrayalCapabilities section:

```
<?xml version="1.0" encoding="UTF-8"?>
<wvs:WVS_Capabilities xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:wvs="http://www.opengis.net/wvs/0.6.0"
    xsi:schemaLocation="http://www.opengis.net/wvs/0.6.0 ../wvsCapabilities.xsd"
xmlns:ows="http://www.opengis.net/ows/1.1"
    xmlns:xlink="http://www.w3.org/1999/xlink" version="0.6.0">
    <ows:ServiceIdentification>
        <ows:ServiceType>WVS</ows:ServiceType>
        <ows:ServiceTypeVersion>0.6.0</ows:ServiceTypeVersion>
    </ows:ServiceIdentification>
    <ows:ServiceProvider>
        <ows:ProviderName>Hasso-Plattner-Institute</ows:ProviderName>
        <ows:ServiceContact/>
    </ows:ServiceProvider>
    <ows:OperationsMetadata>
        <ows:Operation name="GetCapabilities">
            <ows:DCP>
                <ows:HTTP>
                    <ows:Get xlink:href="http://todo/wvs?"/>
                </ows:HTTP>
            </ows:DCP>
        </ows:Operation>
        <ows:Operation name="GetView">
            <ows:DCP>
                <ows:HTTP>
                    <ows:Get xlink:href="http://todo/wvs?"/>
                </ows:HTTP>
            </ows:DCP>
        </ows:Operation>
        . . .
    </ows:OperationsMetadata>
    <wvs:Contents>
        . . .
    </wvs:Contents>
    <wvs:PortrayalCapabilities>
        <wvs:AvailableImageLayer>
            <ows:Identifier>COLOR</ows:Identifier>
            <wvs:AvailableFormat>image/jpeg</wvs:AvailableFormat>
            <wvs:AvailableFormat>image/png</wvs:AvailableFormat>
        </wvs:AvailableImageLayer>
        <wvs:AvailableImageLayer>
            <ows:Identifier>DIFFUSE</ows:Identifier>
            <wvs:AvailableFormat>image/jpeg</wvs:AvailableFormat>
            <wvs:AvailableFormat>image/png</wvs:AvailableFormat>
        </wvs:AvailableImageLayer>
```

```
<wvs:AvailableImageLayer>
    <ows:Identifier>DEPTH</ows:Identifier>
    <wvs:AvailableFormat>image/png; Mode=32Bit</wvs:AvailableFormat>
</wvs:AvailableImageLayer>
<wvs:AvailableImageLayer>
    <ows:Identifier>OBJECTID</ows:Identifier>
    <wvs:AvailableFormat>image/jpeg</wvs:AvailableFormat>
    <wvs:AvailableFormat>image/png</wvs:AvailableFormat>
</wvs:AvailableImageLayer>
<wvs:AvailableImageLayer>
    <ows:Identifier>NORMAL</ows:Identifier>
    <wvs:AvailableFormat>image/jpeg</wvs:AvailableFormat>
    <wvs:AvailableFormat>image/png; Mode=24Bit</wvs:AvailableFormat>
</wvs:AvailableImageLayer>
<wvs:AvailableImageLayer>
    <ows:Identifier>MASK</ows:Identifier>
    <wvs:AvailableFormat>image/jpg</wvs:AvailableFormat>
    <wvs:AvailableFormat>image/png; Mode=1Bit</wvs:AvailableFormat>
</wvs:AvailableImageLayer>
<wvs:AvailableProjection>
    <wvs:ProjectionType>PerspectiveProjection</wvs:ProjectionType>
    <wvs:ProjectionParameter name="FOVX" required="false">
        <ows:AnyValue/>
        <ows:DefaultValue>60</ows:DefaultValue>
        <ows:DataType>double</ows:DataType>
    </wvs:ProjectionParameter>
    <wvs:ProjectionParameter name="Fovy" required="true">
        <ows:AnyValue/>
        <ows:DefaultValue>60</ows:DefaultValue>
        <ows:DataType>double</ows:DataType>
    </wvs:ProjectionParameter>
</wvs:AvailableProjection>
<wvs:AvailableProjection>
    <wvs:ProjectionType>OrthographicProjection</wvs:ProjectionType>
    <wvs:ProjectionParameter name="Left">
        <ows:AnyValue/>
        <ows:DataType>double</ows:DataType>
    </wvs:ProjectionParameter>
    <wvs:ProjectionParameter name="Right">
        <ows:AnyValue/>
        <ows:DataType>double</ows:DataType>
    </wvs:ProjectionParameter>
    <wvs:ProjectionParameter name="Top">
        <ows:AnyValue/>
        <ows:DataType>double</ows:DataType>
    </wvs:ProjectionParameter>
    <wvs:ProjectionParameter name="Bottom">
        <ows:AnyValue/>
        <ows:DefaultValue>123</ows:DefaultValue>
        <ows:DataType>double</ows:DataType>
    </wvs:ProjectionParameter>
</wvs:AvailableProjection>
<wvs:NearPlaneHint>2</wvs:NearPlaneHint>
<wvs:FarPlaneHint>2000.0</wvs:FarPlaneHint>
<wvs:AvailableImageStyle>
    <ows:Abstract>Abstract, non-photo realistic illustration</ows:Abstract>
    <ows:Identifier>NPR</ows:Identifier>
</wvs:AvailableImageStyle>
<wvs:ViewpointHint>
    <ows:Title>Test</ows:Title>
    <ows:Abstract>Test</ows:Abstract>
    <ows:Keywords>
        <ows:Keyword>Test</ows:Keyword>
    </ows:Keywords>
    <ows:Identifier>Test</ows:Identifier>
    <wvs:PerspectiveProjection>
        <wvs:ProjectionType>
            <wvs:POC>100 100 100</wvs:POC>
            <wvs:POI>101 101 101</wvs:POI>
            <wvs:Up>0 0 1</wvs:Up>
            <wvs:FOVX>80</wvs:FOVX>
```

```
                    <wvs:FOVY>60</wvs:FOVY>
                    <wvs:NearPlane>0.0001</wvs:NearPlane>
                    <wvs:FarPlane>10000.0</wvs:FarPlane>
                </wvs:ProjectionType>
            </wvs:PerspectiveProjection>
        </wvs:ViewpointHint>
        <wvs:SupportsMultipleViews>true</wvs:SupportsMultipleViews>
        <wvs:SupportedSpatialSelections>
            <wvs:Method>contains_center</wvs:Method>
            <wvs:Method>intersects</wvs:Method>
            <wvs:Method>cut</wvs:Method>
        </wvs:SupportedSpatialSelections>
        <wvs:AvailableNavigationType>
            <ows:Abstract>Computes new camera position showing at the object pointed at.</ows:Abstract>
            <ows:Identifier>GoTo</ows:Identifier>
            <wvs:MinPositions>1</wvs:MinPositions>
            <wvs:MaxPositions>1</wvs:MaxPositions>
        </wvs:AvailableNavigationType>
    </wvs:PortrayalCapabilities>
</wvs:WVS_Capabilities>
```

Annex E
(informative)

Application scenarios for 3D portrayal

Table 77 — Application scenarios for 3D portrayal

1	**Civil Service**
1.1	**Use a 3D view for processing a building application** A civil service agent can generate a perspective view for a specific address for getting an idea of the current building situation around that address. By picking buildings in the image further building information are displayed.
1.2	**Use perspective views as interface for notifications of claim** A citizen can generate a perspective view for a specific address, can navigate for defining a specific view, can pick a position in that image, and can leave a notification of claim, e.g., about an out-of-order fire hydrant.
1.3	**Show a city planning to the public** For public participation, the municipality of a city can make a city planning available to the public by 3D portrayal services. Via web portal, citizens can view this planning, select and investigate planning alternatives. Either interesting views are offered, or the users can navigate freely.
1.4	**Access city planning data easily** By picking at interesting parts of the perspective view, citizens can request relevant city planning data, e.g., planned size of buildings, planned usage, etc.
1.5	**Allow for commenting a city planning** Citizens can comment a city planning by annotating a specific perspective view by text or even drawing or by selecting and annotating a specific position or object in the view.
1.6	**Inspect underground infrastructure** Engineers and department heads responsible for the maintenance of telecommunication or power cables, sewers, gas pipes, or fresh water pipes can inspect the current situation underground. When a new subways line is planned, the underground infrastructure must be analyzed and checked for conflicts.
2	**Business Development**
2.1	**Present an urban space to an investor** For attracting an investor, a 3D city model can help to investigate interesting urban spaces. It shows the general surrounding, transportation infrastructure, etc. This reduces the number of on-site inspections and, thus, saves time and money.
2.2	**Color buildings according to their usage** For judging the surrounding of an object of interest, the agent defines to color the buildings in the perspective view according to their usage type, e.g., living space in green, public buildings in blue, and industrial buildings in red.
2.3	**Measure distances and areas** The perspective view can support simple analysis functionality. E.g., distances to nearby building areas or the overall size of a specific area could be calculated and displayed.

3	**Real Estate Business**
3.1	**Offering interesting buildings to a customer** An estate agent can use the 3D portrayal services for showing buildings to customers. The perspective views show the appearance of the building, including detailed building structures (doors, windows, balconies, etc.) and real facades. Furthermore the agent can show, describe, and investigate the urban surrounding of these buildings to the potential buyer by help of specific navigation techniques supported by the client application.
3.2	**Highlight buildings to sell** Within an interesting building area, the estate agent can influence the visual appearance of buildings that can be rent or bought. He selects some buildings by their address and others by their owner and defines which color to use for their representation. Furthermore he highlights all nearby public buildings in a different color.
3.3	**Access BIM information easily** For all the buildings in the view, the agent can request additional BIM information such as the age of the building, usage, number of rooms, room sizes, its renovation state, land parcel information, etc. This is done by easily picking the object of interest in the image.
3.4	**Allow for looking inside buildings** Within a city model the agent can select a single building of interest and request a building-specific visualization which allows, for looking into the building and, e.g., see the structure of the rooms at each floor.
3.5	**Calculate and display the visibility of objects** For installing a hotel, an investor looks for a building with a good view, i.e., from where a set of relevant points of interests (e.g., church, market place, monuments, palace garden, etc.) have a high visibility. The system allows the agent to define the points of interest and the area in which to search for suitable buildings. Then, the system calculates the visibility of these POIs for each building, maps this information to the building facades, and generates new perspective views.
4	**Security and Safety**
4.1	**Introduce an operational area to a mobile action force** In preparation for safeguarding a demonstration, action forces need to get an overview of the operation area, including size and topology of street, danger spots, etc. The presentation of the city by 3D portrayal can replace on-site inspections, and thereby reduce cost. If necessary, perspective views can be accessed by mobile devices.
4.2	**Generate views containing escape routes** For a mass event, escape routes have to be defined and must be visualized to the security personnel as well as to the visitors. By help of a styling tool, the safety officer can enrich a perspective view of that area by additional path objects representing these escape routes. This styling description is then published and used by end users.
4.3	**Support Fire Fighters** The fire department requests a detailed indoor building model in case of a fire incident. The 3D indoor model is analyzed in the command center before the fire fighters reach the site. Possibly the visibility in the building is limited due to the smoke. Possible access and escape routes are determined to help the fire fighter evacuating the building.
5	**Tourism**

5.1	**Investigate a vacation spot** A tourist can use web-based 3D portrayal for investigating a vacation spot already from home. For example, this could support the decision for a specific accommodation according to the appearance of the surrounding.
5.2	**Generate a specific touristic 3D city map** For a city tour, a tourist needs specific information such as the location of bus stops, railway stations, hotels, museums, theaters, hotels, bars, restaurants, the tourist information, etc. By an authoring tool, a municipality clerk can create a specific touristic 3D city map containing this information. A styling tool allows him to define how the integrated information shall be represented.
5.3	**Show the current environmental situation** For making a virtual city model more appealing and more realistic, they can include sun and clouds, which could be in accordance with the real weather conditions.
5.4	**Generate a video tour negotiating selected points of interest** For tourists a virtual city tour can give an impression about important points of interest, shopping facilities, etc. By an authoring tool a municipality clerk can define such points of interests, which are used as way points for the virtual tour. The tourist can select some or all of these way points for generating an individual video tour, e.g., a flight or a city walk.
6	**Car and Pedestrian Navigation**
6.1	**Illustrate a track by sequences of perspective views** For illustrating a path description, perspective views along that path can be generated. Even the individual path can be embedded into the visualization. The path illustrations can be derived automatically from the defined path.
6.2	**Guide vehicles and pedestrians on the road.** Mobile users are supported by providing 3D animations or interactive visualizations of the course of the route, which has been calculated by an OpenLS route Service. 3D Visualizations are especially useful for pedestrians using their PDAs, since they can get a better idea of where to go if the system contains also landmarks.

Table 77 lists application scenarios for 3D portrayal for various application areas. This listing is mainly copied from [8].

Annex F
(informative)

Examples of WVS clients

F.1 Introduction

This annex provides application examples of two browser-based WVS clients, which are different in client-side complexity.

F.2 Thin web-based client

Figure F.1 shows a screenshot of a thin browser-based WVS client, which is integrated with a 2D mapping service. The WVS client is JavaScript-based and can be easily integrated into exiting web pages. Facilitating the WVS capabilities for position retrieval (GetPosition), the client allows a user to navigate in the 3D scene by clicking in the image; the arrow indicates desired position and orientation of the virtual camera. Additionally a user can click on features for retrieving thematic information (using the WVS GetFeatureInfo operation), or apply distance measurements (based on the WVS GetMeasurement operation).

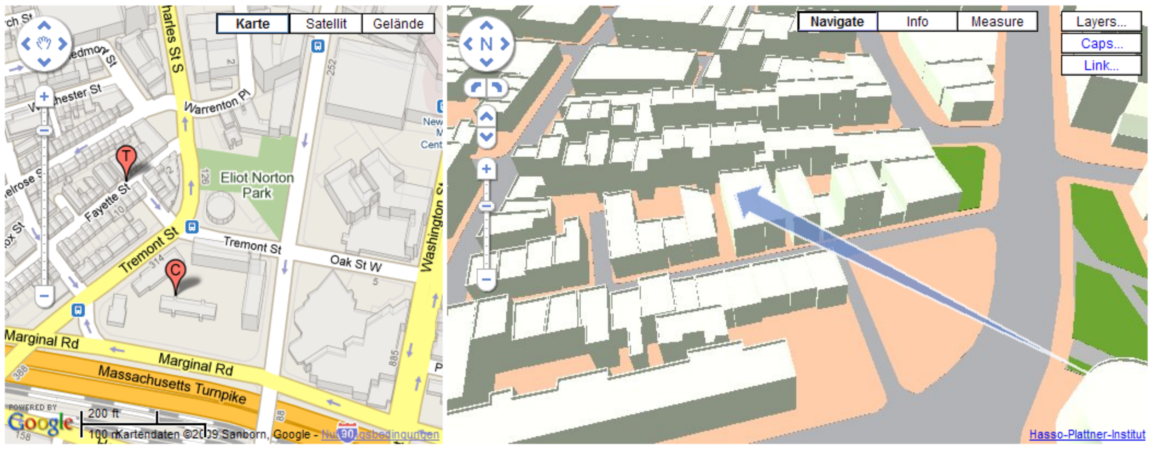


Figure F.1 — Example of a thin WVS client, integrating the 3D representation retrieved from WVS (right) with a 2D map representation requested from a mapping service (left, here: Google maps). 3D data: Boston Redevelopment Auth.

F.3 Medium-size web-based client

Figure F.2 show two screenshots of a medium-size, browser-based WVS client, which is Java-based and can be easily deployed as Java-Applet, e.g., via Java Web Start. This client application retrieves not only COLOR layer from a WVS but also facilitates

OBJECTID and DEPTH information. Object id data is used for rendering purposes such as object selection, depth data is used for computing 3D points at each pixel position for constructing 3D point clouds for spatial analysis functionalities and for rendering purposes and continous real-time navigation in this 3D scene.

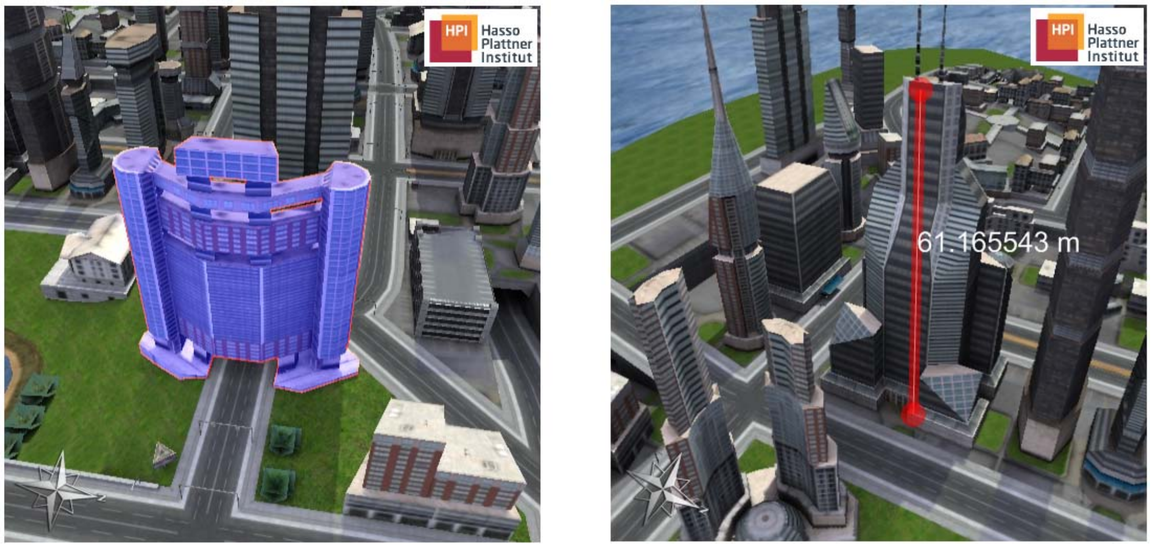


Figure F.2 — Screenshots of a medium-size WVS client, showing object selection functionality (left) and spatial analysis functionalities (right), based only on geometric and thematic data retrieved by the GetView operation.

Bibliography

[1] Guidelines for Successful OGC Interface Standards, OGC document 00-014r1.

[2] Th. Akenine-Möller, E. Haines, N. Hoffman, Real-Time Rendering, 3rd ed. AK Peters, 2008.

[3] T. Saito, T. Takahashi, Comprehensible rendering of 3-D shapes. SIGGRAPH Comput. Graphic., 24(4):197–206, 1990.

[4] A. Watt, F. Policarpo, 3D Games, Real-time Rendering and Software Technology. Addison-Wesley, 2007.

[5] OGC 07-057r2, *OpenGIS Tiled WMS Discussion Paper*, Version 0.3.0, August 2007.

[6] OGC 03-091, *OpenGIS Web Terrain Service*, September 2003.

[7] *OpenGIS Web Perspective View Service (WPVS) Implementation Specification*, Draft proposed version 1.0, October 2005.

[8] OGC 08-140, *3D Portrayal Services – Use Cases*, August 2008.

[9] OGC 09-104, *Draft Web 3D Service Implementation Standard*, proposed Version 0.4.0, July 2009.