

# Open Geospatial Consortium Inc.

Date: 2007-06-29

Reference number of this OGC® document: OGC 05-078r4

Version: 1.1.0 (revision 4)

Category: OGC® Implementation Specification

Editor: Dr. Markus Lupp

## **Styled Layer Descriptor profile of the Web Map Service Implementation Specification**

Copyright © 2007 Open Geospatial Consortium, Inc. All Rights Reserved.  
To obtain additional rights of use, visit <http://www.opengeospatial.org/legal/>.

Document type:	OGC® Implementation Specification
Document subtype:	Profile
Document stage:	Final version
Document language:	English

<b>Contents</b>		<b>Page</b>
1	Scope.....	1
2	Conformance.....	1
3	Normative references.....	1
4	Terms and definitions .....	2
5	Conventions .....	2
5.1	Abbreviated terms .....	2
5.2	UML notation.....	3
6	Web-Map-Server integration .....	4
6.1	A review of WMS 1.3 .....	4
6.2	Styled-Layer Descriptor .....	5
6.3	Web Map Servers and Web Feature/Coverage Servers .....	6
7	GetCapabilities operation (mandatory).....	9
7.1	Introduction .....	9
7.2	GetCapabilities operation request .....	9
7.3	GetCapabilities operation response.....	9
8	DescribeLayer operation (optional).....	10
8.1	Introduction .....	10
8.2	DescribeLayer operation request.....	11
8.2.1	DescribeLayer request parameters.....	11
8.2.2	DescribeLayer request KVP encoding (required).....	12
8.3	DescribeLayer operation response .....	12
8.3.1	Normal response XML encoding.....	13
8.3.2	DescribeLayer exceptions.....	14
9	GetMap operation (mandatory).....	14
9.1	Introduction .....	14
9.2	GetMap operation request .....	14
9.2.1	GetMap request parameters .....	15
9.2.2	GetMap request KVP encoding .....	16
9.2.3	GetMap request XML encoding .....	18
9.3	GetMap operation response.....	19
9.3.1	GetMap exceptions .....	19
10	GetLegendGraphic operation (optional).....	19
10.1	Introduction .....	19
10.2	GetLegendGraphic operation request.....	20
10.2.1	GetLegendGraphic request parameters.....	20
10.2.2	GetLegendGraphic request KVP encoding (required).....	23
10.3	GetLegendGraphic operation response .....	24
10.3.1	Normal response parameters.....	24

10.3.2	GetLegendGraphic exceptions.....	24
10.4	Examples .....	25
11	SLD encoding .....	26
11.1.1	SLD root element.....	26
11.2	Named layers.....	27
11.3	User-defined layers.....	30
11.3.1	Feature Constraints .....	31
11.3.2	Coverage Constraints.....	32
11.4	User-defined styles.....	35
Annex A (normative)	Abstract test suite .....	37
Annex B (normative)	XML schemas.....	38
Annex C (informative)	Example XML documents .....	39

## **i. Preface**

This document explains how the Web Map Server specification can be extended to allow user-defined symbolization of feature and coverage data. It should be read in conjunction with the latest version WMS specification. At the time of writing the latest version WMS specification was defined by the WMS 1.3 Specification.

This document is together with the Symbology Encoding Implementation Specification the direct follow-up of Styled Layer Descriptor Implementation Specification 1.0.0. The old specification document was split up into two document to allow the parts that are not specific to WMS to be reused by other service specifications.

## **ii. Document terms and definitions**

This document uses the specification terms defined in Subclause 5.3 of [OGC 05-008], 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 specification.

## **iii. Submitting organizations**

The following organizations submitted this document to the Open Geospatial Consortium Inc.

CubeWerx Inc.  
lat/lon GmbH (Editor)  
Pennsylvania State University.  
Syncline  
Ionic Software s.a.

#### iv. Document contributor contact points

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

Name	Organization
Larry Bouzane	Compusult Ltd.
Dr. Craig Bruce	CubeWerx Inc.
Ivan Cheung	ESRI
Adrian Cuthbert	m-spatial
Reinhard Erstling	interactive instruments GmbH
Ron Lake	Galdos Systems Inc.
Seb Lessware	Laser-Scan Ltd.
Marwa Mabrouk	ESRI
James Macgill	Google Maps
Dimitri Monie	Ionic Software s.a.
Dr. Markus Lupp	lat/lon GmbH
Dr. Andreas Poth	lat/lon GmbH
Raj Singh	Open Geospatial Consortium
Dan Specht	US Army ERDC
John Vincent	Intergraph Corp.
Peter Vretanos	CubeWerx Inc.

#### v. Revision history

Date	Release	Editor	Primary clauses modified	Description
2001-02-07	01-028	Adrian Cuthbert	initial paper for SLD 0.7.0	WMT-2 Project-Discussion Paper
2001-08-31	01-028r2	Craig Bruce	re-write for SLD 0.7.1	MPP-1 Project-Discussion Paper
2001-11-30	01-028r3	Craig Bruce	update for SLD 0.7.2 and DIPR format	MPP-1.1 DIPR preview
2001-11-30	01-028r4	Craig Bruce	fixed up pre-pages, added GeoSym content	MPP-1.1 DIPR
2001-12-28	01-028r5	Craig Bruce	minor fixes, added 2525B content, example pictures	MPP-1.1 IPR
2002-03-12	02-013	Carl Reed Craig Bruce Bill Lalonde	Modified for submission and consideration as RFC Proposal for SLD Implementation Specification	Implementation Specification
2002-04-24	02-013r1	Bill Lalonde Greg Buehler	Minor formatting changes	Formating for Public Comment
2002-08-15	02-013r2	Craig Bruce	Incorporated RFC changes	Incorporated RFC comments

2004-02-26	02-070r1	Craig Bruce	Incorporated SLD-1.0.20/Style-Management-System changes	First draft for 1.1.0
2004-04-13	02-070r2	Donéa Luc	Incorporated 03-004 change proposal for coverage-data selection and styling	Second draft for 1.1.0
2004-05-01	02-070r3	Clemens Portele, Reinhard Erstling	Incorporated change request 03-095r1, general review for consistency	Third draft for 1.1.0
2004-12-17	02-070r4	Craig Bruce	Partial Incorporation of SLD-RWG & interactive instruments changes; see Annex E.	Fourth draft for 1.1.0
2005-4-11	02-070r5	James Macgill	Completed changes started in r4	Fifth draft for 1.1.0
2005-04-29	02-070r6	Markus Müller, Andreas Poth	Incorporated change request 05-028	Sixth draft for 1.1.0
2005-08-22	02-070r7	Markus Müller, Andreas Poth	Finished changes regarding 05-028	Seventh draft for 1.1.0
2005-10-19	05-078	Markus Müller	All	Split SLD specification in SLD profile for WMS (this document) and Symbology Encoding
2006-09-07	05-078r1	Markus Müller	All	First revision of draft SE 1.1.0 for review by SLD RWG
2006-09-29	05-078r2	Markus Müller	All	Minor changes. Some XML schema errors fixed. Added conformance classes and examples.
2007-01-05	05-078r3	Markus Müller	All	Applied Changes discussed in RWG.. Version for vote on publication by SLD RWG
2007-06-29	05-078r4	Markus Lupp	6.3; Annex B	Changed SRS to CRS in example WMS 1.3 GetMap Request; Adjusted URL to XML schema documents in Annex B

## vi. Changes to the OGC Abstract Specification

The OGC<sup>®</sup> Abstract Specification requires/does not require changes to accommodate the technical contents of this document.

## **Foreword**

This document together with OGC 05-077r4 (Symbology Encoding Implementation Specification) replaces OGC 02-070 and consists of the following part: Styled Layer Descriptor Profile of the Web Map Service Implementation Specification.

This document includes 3 annexes; Annexes A and B are normative, and Annex C is informative.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The OGC shall not be held responsible for identifying any or all such patent rights.

## Introduction

The importance of the visual portrayal of geographic data cannot be overemphasized. The skill that goes into portraying data (whether it be geographic or tabular) is what transforms raw information into an explanatory or decision-support tool. From USGS' topographic map series to NOAA and NIMA's nautical charts to AAA's Triptik, fine-grained control of the graphical representation of data is a fundamental requirement for any professional mapping community.

The current OGC Web Map Service (WMS) specification supports the ability for an information provider to specify very basic styling options by advertising a preset collection of visual portrayals for each available data set. However, while a WMS currently can provide the user with a choice of style options, the WMS can only tell the user the name of each style. It cannot tell the user what portrayal will look like on the map. More importantly, the user has no way of defining their own styling rules. The ability for a human or machine client to define these rules requires a styling language that the client and server can both understand. Defining this language, called the **Symbology Encoding (SE)** is done in a companion document of this specification. This language can be used to portray the output of Web Map Servers, Web Feature Servers and Web Coverage Servers. This document defines how Symbology Encoding can be used in conjunction with Web Map Services. In many cases, however, the client needs some information about the data residing on the remote server before he, she or it can make a sensible request. This led to the definition of new operations for the OGC services (see Clauses 8 and 9) in addition to the definition of the styling language.

There are two basic ways to style a data set. The simplest one is to color all features the same way. For example, one can imagine a layer advertised by a WMS as “hydrography” consisting of lines (rivers and streams) and polygons (lakes, ponds, oceans, etc.). A user might want to tell the server to color the insides of all polygons in a light blue, and color the boundaries of all polygons and all lines in a darker blue. This type of styling requires no knowledge of the attributes or “feature types” of the underlying data, only a language with which to describe these styles. This requirement is addressed by the **FeatureTypeStyle** element in the SE document.

A more complicated requirement is to style features of the data differently depending on some attribute. For example, in a roads data set, style highways with a three-pixel red line; style four-lane roads in a two-pixel black line; and style two-lane roads in a one-pixel black line. Accomplishing this requires the user to be able to find out what attribute of the data set represents the road type. SLD profile of WMS defines the operation that fulfils this need, called **DescribeLayer**. This operation returns the feature types of the layer or layers specified in the request, and the attributes can be discovered with the **DescribeFeatureType** operation of a WFS interface or the **DescribeCoverageType** of a WCS interface.



# Styled Layer Descriptor Profile of the Web Map Service Implementation Specification

## 1 Scope

This OGC<sup>®</sup> Implementation Specification specifies how a Web Map Service can be extended to allow user-defined styling. Different modes for utilizing Symbology Encoding for this purpose are discussed.

## 2 Conformance

Conformance with this specification 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.

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

IETF RFC 2045 (November 1996), *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*, Freed, N. and Borenstein N., eds.,  
<<http://www.ietf.org/rfc/rfc2045.txt>>

IETF RFC 2616 (June 1999), *Hypertext Transfer Protocol – HTTP/1.1*, Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and Berners-Lee, T., eds.,  
<<http://www.ietf.org/rfc/rfc2616.txt>>

IETF RFC 2396 (August 1998), *Uniform Resource Identifiers (URI): Generic Syntax*, Berners-Lee, T., Fielding, N., and Masinter, L., eds.,  
<<http://www.ietf.org/rfc/rfc2396.txt>>

OGC AS 12 (January 2002), *The OpenGIS Abstract Specification Topic 12: OpenGIS Service Architecture (Version 4.3)*, Percivall, G. (ed.),  
<<http://www.opengis.org/techno/abstract/02-112.pdf>>

OGC Adopted Implementation Specification: Web Map Server version 1.3, August 2004, OGC document OGC 04-024, <[http://portal.opengis.org/files/?artifact\\_id=5316](http://portal.opengis.org/files/?artifact_id=5316)>.

OGC Adopted Implementation Specification: Web Feature Service version 1.1, May 2004, OGC document OGC 04-094, <[https://portal.opengeospatial.org/files/?artifact\\_id=8339](https://portal.opengeospatial.org/files/?artifact_id=8339)>.

OGC Adopted Implementation Specification: Filter Encoding version 1.1, May 2004, OGC document OGC 04-095 <[https://portal.opengeospatial.org/files/?artifact\\_id=8340](https://portal.opengeospatial.org/files/?artifact_id=8340)>.

OGC Adopted Implementation Specification: Geography Markup Language version 3.1.1, May 2004, OGC document OGC 04-095 <[https://portal.opengeospatial.org/files/?artifact\\_id=4700](https://portal.opengeospatial.org/files/?artifact_id=4700)>.

OGC Adopted Implementation Specification: Web Coverage Service version 1.0, October 2003, OGC document OGC 03-065r6, <[https://portal.opengeospatial.org/files/?artifact\\_id=3837](https://portal.opengeospatial.org/files/?artifact_id=3837)>.

In addition to this document, this specification includes several normative XML Schema files. Following approval of this document, these schemas will be posted online at the URL <http://schemas.opengeospatial.net/SLD/1.1.0>. These XML Schema files are also bundled with the present document. In the event of a discrepancy between the bundled and online versions of the XML Schema files, the online files shall be considered authoritative.

## 4 Terms and definitions

For the purposes of this specification, the definitions specified in Clause 4 of the OWS Common Implementation Specification [OGC 05-008] shall apply. In addition, the following terms and definitions apply.

### 4.1

#### **map**

Pictorial representation of geographic data

## 5 Conventions

### 5.1 Abbreviated terms

Most of the abbreviated terms listed in Subclause 5.1 of the OWS Common Implementation Specification [OGC 05-008] apply to this document, plus the following abbreviated terms.

GIF	Graphics Interchange Format
JPEG	Joint Photographic Experts Group
PNG	Portable Network Graphics

SVG	Scalable Vector Graphic
WebCGM	Web Computer Graphics Metafile
WCS	Web Coverage Service
WFS	Web Feature Service

## **5.2 UML notation**

Some diagrams that appear in this specification are presented using the Unified Modeling Language (UML) static structure diagram, as described in Subclause 5.2 of [OGC 05-008].

## 6 Web-Map-Server integration

### 6.1 A review of WMS 1.3

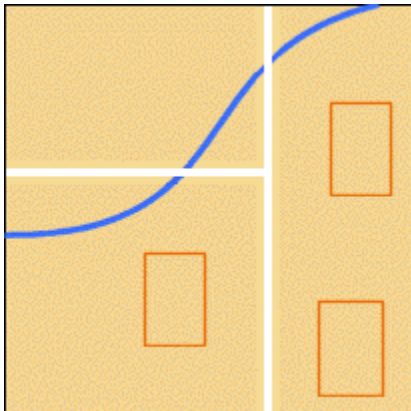
WMS 1.3 and earlier versions describe the appearance of a map in terms of ‘styled layers’. A styled layer can be considered as a transparent sheet with features symbolized upon it. A map is made up of a number of these styled layers put together in a specified order. The styled layers are said to be Z-ordered. Users can define more complex or simpler maps by adding or removing styled layers.

A styled layer itself represents a particular combination of ‘layer’ and a ‘style’ in which that layer can be symbolized. Conceptually, the layer defines a stream of features and the style defines how those features are symbolized. This concept is underlined by the fact that there may be multiple styles in which a layer can be symbolized.

In the WMS specification, the request for a map is encoded as an HTTP-GET or POST request and the appearance for a map portrayal is specified by the **LAYERS** and **STYLES** parameters. Consider the following (incomplete) example map request (which is split over multiple lines for presentation purposes only):

```
http://yourfavoritesite.com/WMS?
  REQUEST=GetMap&
  BBOX=0.0,0.0,1.0,1.0&
  LAYERS=Rivers,Roads,Houses&
  STYLES=CenterLine,CenterLine,Outline
```

Results in the map portrayal shown below:



This is to be interpreted as three ‘styled layers’, namely:

- a) Houses:Outline
- b) Roads:CenterLine
- c) Rivers:CenterLine

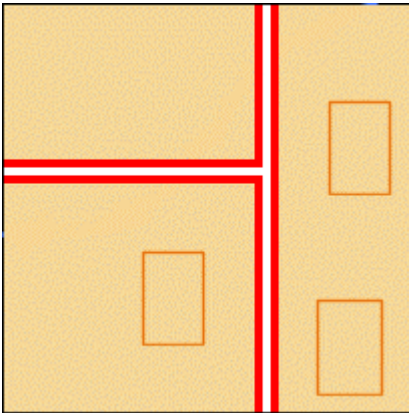
The colon notation is introduced only as a convenience to aid discussion. The **Rivers:CenterLine** styled layer is ‘below’ the **Roads:CenterLine** styled layer, as WMS uses the “painter’s model” and plots each successive layer in the **LAYER** list on top of

the previously rendered layers. Consequently, the roads appear to ‘cross’ the river. It is possible for the same layer to appear more than once, although this is rarely done with the same style.

A common ‘cartographic trick’ to generate what appears to be the boundaries of linear features is to draw them with a thick colored line and then draw them all again with a thinner, lighter line. This is done for the roads in the following (incomplete) map request:

```
http://yourfavoritesite.com/WMS?
REQUEST=GetMap&
BBOX=0.0,0.0,1.0,1.0&
LAYERS=Roads,Roads,Houses&
STYLES=Casing,CenterLine,Outline
```

The resulting map portrayal based upon the above rule is:



This is to be interpreted as three styled layers, namely:

- d) Houses:Outline
- e) Roads:CenterLine
- f) Roads:Casing

It might be noted that the WMS cannot be interrogated for metadata to indicate which styled layers can be meaningfully combined and how. However, a flexible client would allow an end-user to explore the various possibilities.

The WMS 1.3 specification deals with styles and layers which are ‘known’ to the WMS and which are identified by name. For this reason, the rest of this document refers to the layers and styles that have been described above as “named layers” and “named styles”. The WMS specification provides only one way to define a styled layer, as a combination of a named layer and a named style.

## 6.2 Styled-Layer Descriptor

Subclause 6.1 described how the appearance of a map in the WMS specification can be defined as a sequence of styled layers. Styling can also be described using a user-defined XML encoding of a map’s appearance called a Styled-Layer Descriptor (SLD). The SLD format is discussed in detail in Clause 11. Briefly, an SLD includes a

**StyledLayerDescriptor** XML element that contains a sequence of styled-layer definitions. These styled-layer definitions may use named or user-defined layers and named or user-defined styling. Here is a simple SLD that corresponds to the first example from the previous section:

```
<StyledLayerDescriptor version="1.1.0">
  <NamedLayer>
    <Name>Rivers</Name>
    <NamedStyle>
      <Name>CenterLine</Name>
    </NamedStyle>
  </NamedLayer>
  <NamedLayer>
    <Name>Roads</Name>
    <NamedStyle>
      <Name>CenterLine</Name>
    </NamedStyle>
  </NamedLayer>
  <NamedLayer>
    <Name>Houses</Name>
    <NamedStyle>
      <Name>Outline</Name>
    </NamedStyle>
  </NamedLayer>
</StyledLayerDescriptor>
```

The **NamedLayer** and **NamedStyle** elements correspond to the **LAYERS** and **STYLES** of the CGI parameters whereas the “painter’s model” is also used for Z-ordering. An SLD XML document can become much more complex with user-defined styling.

### 6.3 Web Map Servers and Web Feature/Coverage Servers

If a WMS is to symbolize features using a user-defined symbolization, it is necessary to identify the source of the feature data. This specification is designed to permit a wide variety of implementations of WMS that support user-defined symbolization. For example a WMS might symbolize feature or coverage data stored in a remote Web Feature Server (WFS) or Web Coverage Server (WCS), or it might only be able to symbolize data from a specific default feature/coverage store.

In support of this, optional parameters called **REMOTE\_OWS\_TYPE** and **REMOTE\_OWS\_URL** are introduced for HTTP-GET **GetMap** requests that can be used to direct the WMS to a remote WFS, WCS, or other OWS service as the ‘default’ source for feature/coverage data. The presently allowed values for the **REMOTE\_OWS\_TYPE** parameter are “WFS” and “WCS”, though more may be allowed in the future. The **REMOTE\_OWS\_URL** parameter gives the base URL of the service to use. For example, if the URL for a WFS is <http://anothersite.com/WFS?> then the map request from the previous subclause would be converted to look like:

```
http://yourfavoritesite.com/WMS?
  VERSION=1.3.0&
  REQUEST=GetMap&
  CRS=EPSG%3A4326&
  BBOX=0.0,0.0,1.0,1.0&
  SLD=http%3A%2F%2Fmyclientsite.com%2FmySLD.xml&
  WIDTH=400&
```

```

HEIGHT=400&
FORMAT=image/png&
REMOTE_OWS_TYPE=WFS&
REMOTE_OWS_URL=http%3A%2F%2Fanothersite.com%2FWFS%3F

```

This represents the simplest relationship between a WMS and a WFS/WCS. However there is a wide range of possible relationships. To clarify the discussion, this document introduces the concept of ‘component’ and ‘integrated’ servers:

- **Component servers:** these are servers designed to be loosely coupled and work in any combination. For example, a component WMS can symbolize feature/coverage data from any WFS/WCS to which it is directed and GML data that is provided inline. Component servers using feature data are also called Feature Portrayal Services (FPS), while those using coverage data are Coverage Portrayal Services (CPS).
- **Integrated servers:** these are servers that are closely coupled and can only work in particular configurations. For example, an integrated WMS might only be able to symbolize feature/coverage data from the WFS/WCS with which it is integrated.

Whether a particular server is a ‘component’ or ‘integrated’ server says something about how it is implemented. For example a WMS is a ‘valid’ OGC WMS provided it correctly supports the WMS interface. This makes no assumptions about how the WMS is implemented. However, a ‘component’ WMS that can symbolize feature/coverage data only interacts with the data through the WFS/WCS interface, this does say something about the implementation. Of course, it is not important what type of WFS/WCS (component or integrated) that a component WMS is directed to. It is also worth noting that there will continue to be WMSes that can produce maps from sources other than feature data.

There will be a spectrum of WMS with the ability to support user-defined symbolization. This is best illustrated by describing in more detail what might be considered as the ‘two ends’ of the spectrum, represented by a ‘component’ WMS at one end and ‘integrated’ WMS at the other.

- **Component WMS (FPS/CPS)**

Essentially a portrayal engine that can symbolize feature data obtained from one or more remote WFS/WCSes. Typically it has these characteristics:

- A component WMS has no pre-defined ‘named’ layers or styles.
- A component WMS only supports the WMS interface.
- A component WMS can symbolize feature data from any compliant WFS/WCS or GML data provided inline.
- A component WMS supports both user-defined styles and user-defined layers.

- A component WMS that supports WFS as `REMOTE_OWS_TYPE` is called a Feature Portrayal Service, one that supports WCS is a Coverage Portrayal Service.
- Integrated WMS

This is a server representing a closely coupled feature store and a portrayal engine. Typically it has these characteristics:

- An integrated WMS has pre-defined ‘named’ layers and styles.
- An integrated WMS supports the WMS interface and the **GetCapabilities** and **DescribeFeatureType** requests of the WFS interface or the **GetCapabilities** and **DescribeCoverage** requests of the WCS interface.
- An integrated WMS can only symbolize feature data from its own internal feature or coverage data store.
- An integrated WMS only supports user-defined styles being applied to pre-defined ‘named’ layers.

Whether one is using a component or integrated WMS, it must be possible to interrogate (albeit at a relatively superficial level) the underlying feature store. This is because user-defined symbolization makes use of concepts not previously required by WMS. For example, the WMS 1.3 specification makes it possible to interact with a WMS using concepts such as [Named]Layer and [Named]Style but without the need to use concepts such as feature type. By contrast user-defined symbolization needs to be able to define new layers and styles using feature types and feature-type properties. For example, a new layer might be defined as all the features of a particular feature type. This specification seeks to ensure that the bar to creating WMSes that support user-defined symbolization is as low as possible.

The underlying feature/coverage store is interrogated using the WFS/WCS interface. For a component WMS this is not a problem, since the feature/coverage store is indeed a remote WFS/WCS. For an integrated WMS, the server must support both the WMS interface, the **DescribeLayer** operation and a minimal set of WFS/WCS operations. It must support the **DescribeFeatureType** request of a WFS or the **DescribeCoverage** of a WCS. This describes the properties of a feature/coverage type specified by name in the request.

It is also necessary to indicate where a WMS should find the feature data that is to be symbolized. This is done using the following rules:

1. if the SLD specifies a WFS or WCS in the **RemoteOWS** or an **InlineFeature** element of the **UserLayer** element, then the specified data source should be used; otherwise



2. if the GetMap request included CGI **REMOTE\_OWS\_TYPE** and **REMOTE\_OWS\_URL** parameters then that remote service should be used; otherwise
3. the WMS should use a default WFS or WCS.

This approach does not permit features/coverages from different WFS/WCSes to be included in the same styled layer; however, it does allow different styled layers to be based on feature data from different WFS/WCSes. The first two options should only be used for a WMS that can be 'directed' to a remote WFS/WCS. The WMS will advertise this ability in response to a **GetCapabilities** request. For an integrated WMS, the default WFS/WCS is just the one with which it is integrated. However, there is no reason why a component WMS should not have a default WFS/WCS defined.

## 7 GetCapabilities operation (mandatory)

### 7.1 Introduction

The GetCapabilities Request of a SLD-WMS is identical to the one defined by the WMS specification. The response is extended by an element defining the SLD capabilities.

### 7.2 GetCapabilities operation request

See WMS 1.3 for definition of this operation.

### 7.3 GetCapabilities operation response

The response is extended by an element described in the following XML schema fragment:

```
<xsd:element name="UserDefinedSymbolization" substitutionGroup="wms:_ExtendedCapabilities">
  <xsd:complexType>
    <xsd:attribute name="SupportSLD" type="boolean" default="0"/>
    <xsd:attribute name="UserLayer" type="boolean" default="0"/>
    <xsd:attribute name="UserStyle" type="boolean" default="0"/>
    <xsd:attribute name="RemoteWFS" type="boolean" default="0"/>
    <xsd:attribute name="InlineFeatureData" type="boolean" default="0"/>
    <xsd:attribute name="RemoteWCS" type="boolean" default="0"/>
  </xsd:complexType>
</xsd:element>
<xsd:element name="DescribeLayer" type="wms:OperationType"
substitutionGroup="wms:_ExtendedOperation"/>
<xsd:element name="GetLegendGraphic" type="wms:OperationType"
substitutionGroup="wms:_ExtendedOperation"/>
```

**SupportSLD** gives information if the WMS supports any functionality described in this specification. The other four elements give information about the capability of the server to support **UserLayers**, **UserStyles**, Remotes WFS servers, inline Feature (GML) data and remote WCS servers. Additionally the server has to describe its extended operations.

A WMS server which supports SLD has to fulfil either the requirements of an integrated SLD WMS or a component SLD-WMS (or both). An integrated SLD-WMS shall support:

- **UserStyles**
- **DescribeLayer** operation
- either **DescribeFeatureType** or **DescribeCoverage** operation (it may support both)

A component SLD-WMS (regardless if it is of FPS or CPSs type) shall support:

- **UserLayers**
- **UserStyles**

A Feature Portrayal Service shall support

- **RemoteWFS**

A FPS may optionally support **InlineFeatureData**.

A Coverage Portrayal Service shall support

- **RemoteWCS**

The **GetLegendGraphic** operation is optional for all types of SLD-WMS servers

## 8 DescribeLayer operation (optional)

### 8.1 Introduction

Defining a user-defined style requires information about the features being symbolized, or at least their feature/coverage type. Since user-defined styles can be applied to a named layer, there needs to be a mechanism by which a client can obtain feature/coverage-type information for a named layer. This is another example of bridging the gap between the WMS concepts of layers and styles and WFS/WCS concepts such as feature-type and coverage layer. To allow this, a WMS may optionally support the **DescribeLayer** request. This can be applied to multiple layers as shown in the example below:

```
http://yourfavoritesite.com/WMS?  
VERSION=1.3.0&  
REQUEST=DescribeLayer&  
LAYERS=Rivers,Roads,Houses
```

where **DescribeLayer** is a new option for the **REQUEST** parameter and **LAYERS** is the parameter that allows a number of named layers to be specified by name. This is thought to be a better approach than overloading the WMS Capabilities document even more.

The response should be an XML document describing the specified named layers. If any of the named layers are not present, the response is an XML document reporting an exception.

For each named layer, the description should indicate if it is indeed based on feature data and if so it should indicate the WFS/WCS (by a URL prefix) and the feature/coverage types. Note that it is perfectly valid for a named layer not to be describable in this way. Annex B gives the XML Schema for the response.

Only integrated SLD-Web Map Services (of FPS or CPS type) need to support the **DescribeLayer** request.

## 8.2 DescribeLayer operation request

### 8.2.1 DescribeLayer request parameters

A request to perform the DescribeLayer operation shall include the parameters listed and defined in Table 7. 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 “Name” 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 21 in Subclause 9.2.1 of [OGC 05-008].

**Table 1 — Parameters in DescribeLayer operation request**

Name <sup>a</sup>	Definition	Data type and value	Multiplicity and use
service	Service type identifier	Character String type, not empty Value is OWS type abbreviation (“WMS”)	One (mandatory)
request	Operation name	Character String type, not empty (“DescribeLayer”)	One (mandatory)
version	Specification version for operation (WMS)	Character String type, not empty (“1.3.0)	One (mandatory)
layers	names of layers description of requested layers	Character String type, not empty. Multiple layer names separated by ‘,’	Multiple (one is mandatory)
sld_version	Specification version for SLD-specification	Character String type, not empty (“1.1.0)	One (mandatory)
a The name capitalization rules being used here are specified in Subclause 11.6.2 of [OGC 05-008].			

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

### 8.2.2 DescribeLayer request KVP encoding (required)

All SLD-WMS Servers supporting **DescribeLayer** shall implement HTTP GET transfer of the **DescribeLayer** operation request, using KVP encoding. The KVP encoding of the **DescribeLayer** operation request shall use the parameters specified in Table 8. The parameters listed in Table 8 shall be as specified in Table 7 above.

**Table 2 — DescribeLayer operation request URL parameters**

Name and example <sup>a</sup>	Optionality and use	Definition and format
service=WMS	Mandatory	Service type identifier
request= DescribeLayer	Mandatory	Operation name
version=1.3.0	Mandatory	Specification version for this operation
layers=layer_list	Mandatory	names of layers that description is requested for
sld_version=1.1.0	Mandatory	Specification version of supported SLD schema.

<sup>a</sup> 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 05-008].

### 8.3 DescribeLayer operation response

The response should be an XML document describing the specified named layers. If any of the named layers are not present, the response is an XML document reporting an exception.

For each named layer, the description should indicate if it is indeed based on feature/coverage data and if so it should indicate the WFS/WCS (by a URL prefix) and the feature / coverage types. Note that it is perfectly valid for a named layer not to be describable in this way. Annex B gives the XML Schema for the response.

The normal response to a valid **DescribeLayer** shall be an XML document based on the XML scheme defined in Annex B.

**Table 3 — Parts of DescribeLayer operation response**

Name	Definition	Data type and use	Multiplicity and use
Version	Version of SLD schema	String	one (mandatory)
LayerDescription	Describes the data sources for a named layer	complex	Multiple (one is mandatory)

**Table 4 — Parts of LayerDescription element**

Name	Definition	Data type and use	Multiplicity and use
owsType	Type of OWS (either WFS or WCS)	String	one (mandatory)
OnlineResource	Base URL of OWS.	complex	one (mandatory)
Query	Describes FeatureTypeNames or CoverageNames of data used in the described layer	complex	one (mandatory)

### 8.3.1 Normal response XML encoding

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

```
<xsd:schema targetNamespace="http://www.opengis.net/sld" xmlns:sld="http://www.opengis.net/sld"
xmlns:wfs="http://schemas.opengis.net/wfs" xmlns:se="http://www.opengis.net/se"
xmlns:ows="http://www.opengis.net/ows" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
  <xsd:import namespace="http://schemas.opengis.net/wfs" schemaLocation="../../wfs/1.1.0/wfs.xsd"/>
  <xsd:import namespace="http://www.opengis.net/se" schemaLocation="../../se/1.1.0/common.xsd"/>
  <xsd:import namespace="http://www.opengis.net/ows"
schemaLocation="../../ows/1.0.0/owsCommon.xsd"/>
  <xsd:include schemaLocation="StyledLayerDescriptor.xsd"/>
  <xsd:annotation>
    <xsd:documentation>
      Styled Layer Descriptor version 1.1.0 (2006-09-29)
    </xsd:documentation>
  </xsd:annotation>
  <xsd:element name="DescribeLayerResponse" type="sld:DescribeLayerResponseType"/>
  <xsd:complexType name="DescribeLayerResponseType">
    <xsd:sequence>
      <xsd:element name="Version" type="ows:VersionType"/>
      <xsd:element name="LayerDescription" type="sld:LayerDescriptionType"
maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="LayerDescriptionType">
    <xsd:sequence>
      <xsd:element name="owsType" type="sld:owsTypeType"/>
      <xsd:element ref="se:OnlineResource"/>
      <xsd:element name="TypeName" type="sld:TypeNameType" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="TypeNameType">
    <xsd:choice>
      <xsd:element ref="se:FeatureTypeName"/>
    </xsd:choice>
  </xsd:complexType>
</xsd:schema>
```

```

        <xsd:element ref="se:CoverageName"/>
    </xsd:choice>
</xsd:complexType>
<xsd:simpleType name="owsTypeType">
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value="wfs"/>
        <xsd:enumeration value="wcs"/>
    </xsd:restriction>
</xsd:simpleType>
</xsd:schema>

```

### 8.3.2 DescribeLayer exceptions

When an SLD-WMS server encounters an error while performing a DescribeLayer operation, it shall return an exception report message as specified in Subclause 7.4 of [OGC 05-008]. The allowed standard exception codes shall include those listed in Table 10. For each listed exceptionCode, the contents of the “locator” parameter value shall be as specified in the right column of Table 10.

**Table 5 — Exception codes for DescribeLayer 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
NoApplicableCode	No other exceptionCode specified by this service and server applies to this exception	None, omit “locator” parameter

## 9 GetMap operation (mandatory)

### 9.1 Introduction

GetMap is defined in WMS 1.3. The SLD profile for WMS defines additional parameters allowing clients to request layers to be portrayed according to some specified style. The basic WMS specification defines HTTP GET (mandatory) and HTTP POST (optional) for invoking operations.

### 9.2 GetMap operation request

Three approaches are defined to allow a client to take advantage of SLD symbology:

- a) The client interacts with the WMS using HTTP GET but the request can reference a remote SLD.

- b) The client uses the HTTP GET method but includes the SLD XML document in-line with the GET request in an SLD\_BODY CGI parameter (with appropriate character encoding).
- c) The client interacts with the WMS using HTTP POST with the **GetMap** request encoded in XML , as described in section 9.2.3 and including an embedded SLD.

The third method is technically superior but there has been a great lack of vendor support in the past for an XML-POST **GetMap**-request method. Use of the second method, which is a compromise between the first and third methods, can encounter problems resulting from excessively long URLs.

As Symbology Encoding documents (**FeatureTypeStyle** or **CoverageStyle** elements) can also be referenced by xlink in an SLD, the three options described above are only the basic varieties. A client may for example use option a) above while the referenced SLD itself only includes references to remote **FeatureTypeStyles** or **CoverageStyles**. In the other extreme, a client may use option c) creating a complex XML request encompassing the GetMap parameters, with the SLD and Symbology Encodings also included.

It is important to note that in all cases the WMS has no prior knowledge of the SLD contents. There is a wide spectrum of possible clients. Some may allow a user to switch between a number of predefined maps, each specified by its own pre-defined SLD. Others may allow a user to interactively define how they wish a map to appear and construct the necessary SLD ‘on-the-fly’. All of the approaches described above allow a client application to do this but the first one requires that the client is to place the SLD document in a Web location accessible to the WMS.

### 9.2.1 GetMap request parameters

SLD introduces four additional parameters that can be used in a GetMap request.

**Table 6 — Parameters in GetMap operation request**

Name <sup>a</sup>	Definition	Data type and value	Multiplicity and use
sld   sld_body   StyledLayerDescriptor	Styled Layer Descriptor. Three variants can be used, depending on the actual encoding	complex type	optional
remote_ows_type	Type of OWS	Character String type, not empty (“WFS” or “WCS”)	optional
remote_ows_url	base URL of OWS	URL (“http://www.server.com/wms”)	optional
sld_version	Specification version for SLD-specification	Character String type, not empty (“1.1.0”)	One (mandatory)
a The name capitalization rules being used here are specified in Subclause 11.6.2 of [OGC 05-008].			

Besides these additional parameters, the GetMap-request may differ in one other aspect to the standard GetMap: the **LAYERS** parameter is made optional, as it is not needed in case of GetMap requests to component SLD-WMS.

### 9.2.2 GetMap request KVP encoding

Consider the example (incomplete) **GetMap** request from the introductory section:

```
http://yourfavoritesite.com/WMS?
  REQUEST=GetMap&
  BBOX=0.0,0.0,1.0,1.0&
  LAYERS=Rivers,Roads,Houses&
  STYLES=CenterLine,CenterLine,Outline&
  WIDTH=400&
  HEIGHT=400&
  FORMAT=image/png
```

It has already been described, in Subclause 6.3, how the **LAYERS** and **STYLES** parameters could be encoded in an SLD. The request references the SLD using a **SLD** parameter, which replaces the **LAYERS** and **STYLES** parameters. The SLD itself must be accessible to the WMS and is identified using a URL. The URL must be encoded prior to inclusion as a parameter value, just as layer and style names are already encoded.

Assuming the URL for the prepared SLD document is

**http://myclientsite.com/mySLD.xml** then the above map request would be converted to look like:

```
http://yourfavoritesite.com/WMS?
  REQUEST=GetMap&
  BBOX=0.0,0.0,1.0,1.0&
  SLD=http%3A%2F%2Fmyclientsite.com%2FmySLD.xml&
  WIDTH=400&
  HEIGHT=400&
  FORMAT=PNG
```

The prepared SLD document for this example would have the content of the **StyledLayerDescriptor** example from Subclause 6.2, with appropriate standard XML header tags. The SLD document could also be included in-line with the GET request as in the following long example (which does not include schema or namespace references):

```
http://yourfavoritesite.com/WMS?
  REQUEST=GetMap&
  BBOX=0.0,0.0,1.0,1.0&
  SLD_BODY=%3C%3Fxml+version%3D%221.0%22+encoding%3D%22UTF-8%22%3F%3E%3CStyled
  LayerDescriptor+version%3D%221.1.0%22%3E%3CNamedLayer%3E%3CName%3ERivers%3C%2FN
  ame%3E%3CNamedStyle%3E%3CName%3ECenterLine%3C%2FName%3E%3C%2FNamedStyle%3E%3C%2
  FNamedLayer%3E%3CNamedLayer%3E%3CName%3ERoads%3C%2FName%3E%3CNamedStyle%3E%3CNa
  me%3ECenterLine%3C%2FName%3E%3C%2FNamedStyle%3E%3C%2FNamedLayer%3E%3CNamedLayer
  %3E%3CName%3EHouses%3C%2FName%3E%3CNamedStyle%3E%3CName%3EOutline%3C%2FName%3E%
  3C%2FNamedStyle%3E%3C%2FNamedLayer%3E%3C%2FStyledLayerDescriptor%3E
  WIDTH=400&
  HEIGHT=400&
  FORMAT=PNG
```

There may be other complications in addition to the excessively long URLs with this approach if UTF-8 characters outside of the 7-bit ASCII range are used, as HTTP is defined to use the ISO Latin-1 character set. The advantages are that the client does not



need to publish the SLD document on the Web and simple clients that are unable to use the POST method can use this method.

**Table 7 — Additional GetMap operation request URL parameters**

Name and example <sup>a</sup>	Optionality and use	Definition and format
sld=http://www.server.com/sld.xml	Optional (either use this parameter or SLD_BODY)	URL pointing to a StyledLayerDescriptor XML file
sld_body= %3C%3Fxml+version%3D%221.0%22+encoding%3D%22UTF-8%22%3F%3E%3CStyledLayerDescriptor+version%3D%221.1.0%22%3E%3CNamedLayer%3E%3CName%3ERivers%3C%2FName%3E%3CNamedStyle%3E%3CName%3ECenterLine%3C%2FName%3E%3C%2FNamedStyle%3E%3C%2FNamedLayer%3E%3CNamedLayer%3E%3CName%3ERoads%3C%2FName%3E%3CNamedStyle%3E%3CName%3ECenterLine%3C%2FName%3E%3C%2FNamedStyle%3E%3C%2FNamedLayer%3E%3CName%3EHouses%3C%2FName%3E%3CNamedStyle%3E%3CName%3EOutline%3C%2FName%3E%3C%2FNamedStyle%3E%3C%2FNamedLayer%3E%3C%2FStyledLayerDescriptor%3E	Optional (either use this parameter or SLD)	URL-encoded StyledLayerDescriptor
remote_ows_type=wfs	Optional	Type of OWS (wfs or wcs)
remote_ows_url=http://some.server.com/wfs	Optional	Base URL of an OWS
sld_version=1.1.0	Mandatory	Specification version of supported SLD schema.
<p><sup>a</sup> 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 05-008].</p>		

To make the HTTP-GET methods more practical for use, the SLD can also be used in one of two different modes depending on whether the **LAYERS** parameter is present in the request. If it is not present, then all layers identified in the SLD document are rendered with all defined styles, which is equivalent to the XML-POST method of usage. If the **LAYERS** parameter is present, then only the layers identified by that parameter are rendered and the SLD is used as a “style library”.

When an SLD is used as a style library, the **STYLES** CGI parameter is interpreted in the usual way in the **GetMap** request, except that the handling of the style names is organized so that the styles defined in the SLD take precedence over the named styles

stored within the map server. The user-defined SLD styles can be given names and they can be marked as being the default style for a layer. To be more specific, if a style named “**CenterLine**” is referenced for a layer and a style with that name is defined for the corresponding layer in the SLD, then the SLD style definition is used. Otherwise, the standard named-style mechanism built into the map server is used. If the use of a default style is specified and a style is marked as being the default for the corresponding layer in the SLD, then the default style from the SLD is used; otherwise, the standard default style in the map server is used.

### 9.2.3 GetMap request XML encoding

The alternative approach is to communicate with the WMS using HTTP POST with SLD as a component of the WMS GetMap request. Unfortunately, recent WMS specifications have not included a POST encoding, so it is necessary to define it in this specification.

Using the POST **GetMap** method, the running example translates into the following XML encoding:

```
<GetMap xmlns="http://www.opengis.net/sld" xmlns:gml="http://www.opengis.net/gml"
xmlns:ogc="http://www.opengis.net/ogc" xmlns:ows="http://www.opengis.net/ows"
xmlns:se="http://www.opengis.net/se" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/sld GetMap.xsd" version="1.3.0">
  <StyledLayerDescriptor version="1.1.0">
    <NamedLayer>
      <se:Name>Rivers</se:Name>
      <NamedStyle>
        <se:Name>CenterLine</se:Name>
      </NamedStyle>
    </NamedLayer>
    <NamedLayer>
      <se:Name>Roads</se:Name>
      <NamedStyle>
        <se:Name>CenterLine</se:Name>
      </NamedStyle>
    </NamedLayer>
    <NamedLayer>
      <se:Name>Houses</se:Name>
      <NamedStyle>
        <se:Name>Outline</se:Name>
      </NamedStyle>
    </NamedLayer>
  </StyledLayerDescriptor>
  <CRS>EPSG:4326</CRS>
  <BoundingBox crs="http://www.opengis.net/gml/srs/epsg.xml#4326">
    <ows:LowerCorner>-180.0 -90.0</ows:LowerCorner>
    <ows:UpperCorner>180.0 90.0</ows:UpperCorner>
  </BoundingBox>
  <Output>
    <Size>
      <Width>1024</Width>
      <Height>512</Height>
    </Size>
    <wms:Format>image/jpeg</wms:Format>
    <Transparent>>false</Transparent>
  </Output>
  <Exceptions>XML</Exceptions>
</GetMap>
```

Although this example describes how a WMS specification request can be converted to use an SLD, the mechanism can be used with any valid SLD. Specifically it can be used with an SLD that includes user-defined symbolization.

### 9.3 GetMap operation response

The response to a GetMap operation is a graphic file as described in the WMS specification.

#### 9.3.1 GetMap exceptions




Exception codes are described in WMS 1.3.0 specification.

## 10 GetLegendGraphic operation (optional)

### 10.1 Introduction

Legends are normally included with maps to indicate to the user how various features are represented in the map. It is therefore important to be able to produce a legend on a map-display client for styles that are represented in SLD / Symbology Encoding format.

The structuring of SLD **UserStyles** into SE **FeatureTypeStyles/CoverageStyles** and **Rules** provides convenient packaging for this purpose, since rules identify each different kind of graphic symbolization that may be present in a map. Given the information in an SLD **UserStyle**, a map-viewer client could generate a legend entry for a layer in the following format:

<b>Roads Layer</b>	
	<b>Highways</b>
	<b>Collector Roads</b>
	<b>Minor Roads</b>

The icon symbols are graphics (images) showing how the rule is rendered. Abstracts or conditions could be displayed by clicking on the titles, etc. The exact presentation of this information is at the discretion of the viewer client.

Generating this kind of display may involve a significant amount of processing on the client. The client will need to examine the selected SLD style and determine which rules apply at the currently used map scale. Then, it will generate something analogous to the above 'form' with the Layer and Rule titles in HTML Java Swing or whatever the environment, using references for the images. Alternatively, the job of producing a meaningful legend entry for a style could be passed on to the server side in a similar way to how it is done now with WMS **LegendURLs**.

The image references make use of the **GetLegendGraphic** operation of the SLD-WMS interface, with a separate reference for each image. The parameterization of the operation needs to be “overloaded” in the same way that parameters for the **GetMap** with an SLD are overloaded in order to handle the different kinds of clients. I.e., there is an XML-based HTTP-POST method for executing **GetMap**, and there are HTTP-GET methods using **SLD=** and **SLD\_BODY=** parameters to reference/transport the SLD for use in either “literal” or “library” mode (depending on whether the **LAYERS=** parameter is present).

The **GetLegendGraphic** operation itself is optional for an SLD-enabled WMS. It provides a general mechanism for acquiring legend symbols, beyond the **LegendURL** reference of WMS Capabilities. Servers supporting the **GetLegendGraphic** call might code **LegendURL** references as **GetLegendGraphic** for interface consistency. Vendor-specific parameters may be added to **GetLegendGraphic** requests and all of the usual OGC-interface options and rules apply. No XML-POST method for **GetLegendGraphic** is presently defined.

An alternative approach to using a **GetLegendGraphic** operation would be for the viewer client to render a style sample directly itself using the style description. This would save some interactions between the client and server and would allow the viewer client to present consistent sample shapes (across remote map servers from different vendors), although the legend graphics might look different from the graphics actually rendered in the map since the viewer and server may have different rendering engines and different graphical capabilities.

The **LegendGraphic** element of an **SE Rule** (defined in Symbology Encoding) actually only has a limited role in building legends. For vector types, a map server would normally render a standard vector geometry (such as a box) with the given symbolization for a rule. But for some layers, such as for Digital Elevation Model (DEM) data, there is not really a “standard” geometry that can be rendered in order to get a good representative image. So, this is what the **LegendGraphic** SE element is intended for, to provide a substitute representative image for a **Rule**. For example, it might reference a remote URL for a DEM layer called “**GTOPO30**”:

```
http://www.vendor.com/sld/icons/COLORMAP_GTOPO30.png
```

## 10.2 GetLegendGraphic operation request

### 10.2.1 GetLegendGraphic request parameters

A request to perform the **GetLegendGraphic** operation shall include the parameters listed and defined in Table 8. 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 “Name” 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 21 in Subclause 9.2.1 of [OGC 05-008].

**Table 8 — Parameters in GetLegendGraphic operation request**

<b>Name</b> <sup>a</sup>	<b>Definition</b>	<b>Data type and value</b>	<b>Multiplicity and use</b>
service	Service type identifier	Character String type, not empty Value is OWS type abbreviation (“WMS”)	One (mandatory)
request	Operation name	Character String type, not empty (“GetLegendGraphic”)	One (mandatory)
version	Specification version for operation	Character String type, not empty (“1.3.0”)	One (mandatory)
layer	Layer for which to produce legend graphic.	Character String type, not empty.	One (mandatory)
style	Style of layer for which to produce legend graphic. If not present, the default style is selected. The style may be any valid style available for a layer, including non-SLD internally-defined styles.	Character String type, not empty.	Optional
remote_ows_type	Type of OWS	Character String type, not empty (“WFS” or “WCS”)	optional
remote_ows_url	base URL of OWS	URL (“http://www.server.com/wfs”)	optional
featuretype	Feature type for which to produce the legend graphic. This is not needed if the layer has only a single feature type.	QName type, not empty.	Optional. A request can only obtain a featuretype or coverage parameter.
coverage	Coverage for which to produce the legend graphic. This is not needed if the layer has only a single coverage.	Character String type, not empty.	Optional. A request can only obtain a featuretype or coverage parameter.
rule	Rule of style to produce legend graphic for, if applicable. In the case that a style has multiple rules but no specific rule is selected, then the map server is obligated to produce a graphic that is representative of all of the rules of the style.	Character String type, not empty.	Optional
scale	In the case that a rule is not specified for a style, this parameter may assist the server in selecting a more appropriate	Double	Optional

	representative graphic by eliminating internal rules that are out-of-scope. This value is a standardized scale denominator, defined in Symbology Encoding.		
sld	This parameter specifies a reference to an external SLD document. It works in the same way as the SLD= parameter of the WMS GetMap operation.	Character String type, not empty.	Optional
sld_body	This parameter allows an SLD document to be included directly in an HTTP-GET request. It works in the same way as the SLD_BODY= parameter of the WMS GetMap operation.	<StyledLayerDescriptor>	Optional
format	This gives the MIME type of the file format in which to return the legend graphic. Allowed values are the same as for the FORMAT= parameter of the WMS GetMap request.	Character String type, not empty.	Required
width	This gives a hint for the width of the returned graphic in pixels. Vector-graphics can use this value as a hint for the level of detail to include.	integer	Optional
height	This gives a hint for the height of the returned graphic in pixels.	integer	Optional
exceptions	This gives the MIME type of the format in which to return exceptions. Allowed values are the same as for the EXCEPTIONS= parameter of the WMS GetMap request.	Character String type, not empty.	Optional
sld_version	Specification version for SLD-specification	Character String type, not empty ("1.1.0")	One (mandatory)
a The name capitalization rules being used here are specified in Subclause 11.6.2 of [OGC 05-008].			

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

### 10.2.2 GetLegendGraphic request KVP encoding (required)

Servers can 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 9. The parameters listed in Table 8 shall be as specified in Table 8 above.

**Table 9 —GetLegendGraphic operation request URL parameters**

Name and example <sup>a</sup>	Optionality and use	Definition and format
service=WMS	Mandatory	Service type identifier
request= GetLegendGraphic	Mandatory	Operation name
version=1.1.0	Mandatory	Specification and schema version for this operation
layer=lakes	Mandatory	name of layer
style=blue	Optional	name of style
namespace=xmlns(myns=http://www.someserver.com)	Mandatory if featurtype parameters is used	Identifier of namespace of FeatureType parameter as defined in WFS specification
remote_ows_type=wfs	Optional	Type of OWS (wfs or wcs)
remote_ows_url=http://someserver.com/sld-wms	Optional	Base URL of an OWS
featurtype= myns:lakes	Optional	Reference to feature TypeName to be used to retrieve desired features from WFS
coverage=myns:dem	Optional	Reference to coverage name to be used to retrieve desired features from WFS

rule=biglakes	Optional	name of rule
scale	Optional	standardized scale denominator as defined in Symbology Encoding
sld	Optional	URL reference to a SLD document
sld_body	Optional	Inline SLD document
Format=image/png	Optional, include when multiple output formats available and desired format other than specified default, if any	MIME type of format in which output data should be encoded
Width=300	Optional	Positive integer pixel width of desired output
Height=300	Optional	Positive integer pixel height of desired output
Exceptions=BLANK	Optional, include when default XML not desired	Identifier of format in which operation exceptions should be returned
sld_version=1.1.0	Mandatory	Specification version of supported SLD schema.
<p>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 05-008].</p>		

### 10.3 GetLegendGraphic operation response

#### 10.3.1 Normal response parameters

The normal response to a valid GetLegendGraphic operation request shall be a picture.

#### 10.3.2 GetLegendGraphic exceptions

When an SLD-WMS server encounters an error while performing a GetLegendGraphic operation, it shall return an exception report message as specified in Subclause 7.4 of [OGC 05-008]. The allowed standard exception codes shall include those listed in Table 10. For each listed exceptionCode, the contents of the “locator” parameter value shall be as specified in the right column of Table 10.

NOTE To reduce the need for readers to refer to other documents, the first four values listed below are copied from Table 20 in Subclause 8.3 of [OGC 05-008].

**Table 10 — 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
NoApplicableCode	No other exceptionCode specified by this service and server applies to this exception	None, omit “locator” parameter



## 10.4 Examples

An SLD-WMS operation request for GetLegendGraphic can look like this encoded in KVP:

```
http://www.vendor.com/wms.cgi?  
VERSION=1.1.0&  
REQUEST=GetLegendGraphic&  
LAYER=ROADL_1M%3Alocal_data&  
STYLE=my_style&  
RULE=highways&  
SLD=http%3A%2F%2Fwww.sld.com%2Fstyles%2Fkpp01.xml&  
WIDTH=16&  
HEIGHT=16&  
FORMAT=image%2Fgif&
```

which would produce a 16x16 icon for the **Rule** named “**highways**” defined within layer “**ROADL\_1M:local\_data**” in the SLD. The list of available formats for legend graphics and exceptions can be assumed to be the same as are available for a map in the WMS **GetMap** request.

## 11 SLD encoding

The WMS-layers level of SLD is defined in the “StyledLayerDescriptor.xsd” XML-Schema file and provides the “glue” between feature styling as defined by Symbology Encoding and WMS layers. This level of definitions has been decoupled from the feature-style and symbol definitions to make it convenient to perform feature styling in environments other than inside of a WMS.

### 11.1.1 SLD root element

An SLD document is defined as a sequence of styled layers. The root **StyledLayerDescriptor** is defined by the following XML-Schema fragments:

```
<xsd:element name="StyledLayerDescriptor">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="se:Name" minOccurs="0"/>
      <xsd:element ref="se:Description" minOccurs="0"/>
      <xsd:element ref="sld:UseSLDLibrary" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
        <xsd:element ref="sld:NamedLayer"/>
        <xsd:element ref="sld:UserLayer"/>
      </xsd:choice>
    </xsd:sequence>
    <xsd:attribute name="version" type="se:VersionType" use="required"/>
  </xsd:complexType>
</xsd:element>

<xsd:element name="UseSLDLibrary">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="se:OnlineResource"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

The elements **OnlineResource** and **VersionType** are part of Symbology Encoding and described there. The **version** attribute gives the SLD version of an SLD document, to facilitate backward compatibility with static documents stored in various different versions of the SLD specification. The string has the format “x.y.z”, the same as in other OGC Implementation specifications. For example, an SLD document stored according to this specification would have the version string “1.1.0”. The attribute is required.

Note that version numbers are used differently with SLD from how they are with OGC Web services like WMS. Version negotiation cannot be performed in the same way, since SLD is not a service. Instead, the SLD versions supported must either be implied by the web service they are associated with, or the web service must give an explicit list of supported SLD versions in its capabilities. This issue is unresolved.

The **Name** element allows a symbolic name to be associated with a given SLD document. This element is used with most “objects” defined by SE and SLD to allow them to be referenced. Names must be unique in the context in which they are defined.

The **Description** element is also reused throughout SE and SLD and gives an informative description of the “object” being defined. This information can be extracted and used for such purposes as creating informal searchable metadata in catalogue systems. More metadata fields may be added to this element in the future. The **Name** is not considered to be part of a description since a name has a functional use that is more than just descriptive.

The **UseSLDLibrary** element provides the ability of handling external SLD documents to be used in library-mode even when using XML-encoded POST requests with a WMS. (The library mode can be accessed with the HTTP-GET method by supplying an **SLD** CGI parameter in addition to **LAYERS** and **STYLES** CGI parameters.) This addition merely exercises pre-existing functionality in WMS, but it does add the wrinkle of making SLD-library references iterative and (syntactically) recursive. Successive definitions are applied “on top of” previous ones to determine the scoping of overlapping style definitions. The **OnlineResource** must refer to an SLD document.

The styled layers can correspond to either named layers (**NamedLayer**) or user-defined layers (**UserLayer**), which are described in subsequent subclauses. There may be any number of either type of styled layer, including zero, mixed in any order. The order that the layer references appear in the SLD document will be the order that the styled layers are rendered, with successive styled layers rendered on top of previous styled layers.

## 11.2 Named layers

A “layer” is defined as a collection of features that can be potentially of various mixed feature types. A named layer is a layer that can be accessed from an OGC Web Server using a well-known name. For example, the WMS interface uses the **LAYER** CGI parameter to reference named layers as in the example parameter from Clause 6 of:

```
LAYERS=Rivers , Roads , Houses
```

The equivalent named-layer specification mechanism in SLD is defined by the following XML-Schema fragment:

```
<xsd:element name="NamedLayer">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="se:Name"/>
      <xsd:element ref="se:Description" minOccurs="0"/>
      <xsd:element ref="sld:LayerFeatureConstraints" minOccurs="0"/>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
        <xsd:element ref="sld:NamedStyle"/>
        <xsd:element ref="sld:UserStyle"/>
      </xsd:choice>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

The **Name** and **Description** elements are common to most SLD “objects”. The **Name** identifies the well-known name of the layer being referenced, and is required. All possible well-known names are usually identified in the capabilities document for a server. The **Description** is informative.

The **LayerFeatureConstraints** element is optional in a **NamedLayer** and allows the user to specify constraints on what features of what feature types are to be selected by the named-layer reference. It is essentially a filter that allows the selection of fewer features than are present in the named layer. This element is discussed in Subclause 11.3 where it is more apropos.

A named styled layer can include any number of named styles and user-defined styles, including zero, mixed in any order. If zero styles are specified, then the default styling for the specified named layer is to be used.

A named style, similar to a named layer, is referenced by a well-known name. A particular named style only has meaning when used in conjunction with a particular named layer. All available styles for each available layer are normally named in a capabilities document.

The WMS interface uses the **STYLES** CGI parameter to reference named styles relative to named **LAYERS**, as in the following example parameters:

```
LAYERS=Rivers , Roads , Houses &  
STYLES=CenterLine , CenterLine , Outline
```

Parallel lists of corresponding layer names and style names are used in the CGI interface because the CGI interface is not powerful enough to properly nest the named-style references within the named-layer references. However, the SLD mechanism is. The equivalent named-style selection mechanism in SLD is defined by the following DTD fragment:

```
<xsd:element name="NamedStyle">  
  <xsd:complexType>  
    <xsd:sequence>  
      <xsd:element ref="se:Name"/>  
      <xsd:element ref="se:Description" minOccurs="0"/>  
    </xsd:sequence>  
  </xsd:complexType>  
</xsd:element>
```

The **Name** element simply identifies the well-known style name. The **Description** is informative.

The SLD example corresponding to the example WMS parameters above is:

```
<StyledLayerDescriptor version="1.1.0">  
  <NamedLayer>  
    <Name>Rivers</Name>  
    <NamedStyle>  
      <Name>CenterLine</Name>  
    </NamedStyle>  
  </NamedLayer>  
</StyledLayerDescriptor>
```

```

</NamedLayer>
<NamedLayer>
  <Name>Roads</Name>
  <NamedStyle>
    <Name>CenterLine</Name>
  </NamedStyle>
</NamedLayer>
<NamedLayer>
  <Name>Houses</Name>
  <NamedStyle>
    <Name>Outline</Name>
  </NamedStyle>
</NamedLayer>
</StyledLayerDescriptor>

```

Similarly a ‘cased’ roads example of:

```

LAYERS=Roads , Roads , Houses&
STYLES=Casing , CenterLine , Outline

```

becomes:

```

<StyledLayerDescriptor version="1.1.0">
  <NamedLayer>
    <Name>Roads</Name>
    <NamedStyle>
      <Name>Casing</Name>
    </NamedStyle>
  </NamedLayer>
  <NamedLayer>
    <Name>Roads</Name>
    <NamedStyle>
      <Name>CenterLine</Name>
    </NamedStyle>
  </NamedLayer>
  <NamedLayer>
    <Name>Houses</Name>
    <NamedStyle>
      <Name>Outline</Name>
    </NamedStyle>
  </NamedLayer>
</StyledLayerDescriptor>

```

However, this can be encoded in an alternative manner which does not require the repeated definition of the **NamedLayer** with name “**Roads**”, since each **NamedLayer** may include any number of style references:

```

<StyledLayerDescriptor version="1.1.0">
  <NamedLayer>
    <Name>Roads</Name>
    <NamedStyle>
      <Name>Casing</Name>
    </NamedStyle>
    <NamedStyle>
      <Name>CenterLine</Name>
    </NamedStyle>
  </NamedLayer>
  <NamedLayer>
    <Name>Houses</Name>

```

```

    <NamedStyle>
      <Name>Outline</Name>
    </NamedStyle>
  </NamedLayer>
</StyledLayerDescriptor>

```

Note that the above SLD defines three styled layers, even though there are only two **NamedLayer** elements.

User-defined styles (**UserStyle** in the Schema) are discussed in Clause 11.4.

### 11.3 User-defined layers

In addition to using named layers, it is also useful to be able to define custom user-defined layers for rendering. The Schema fragment for user-defined layers is as follows:

```

<xsd:element name="UserLayer">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="se:Name" minOccurs="0"/>
      <xsd:element ref="se:Description" minOccurs="0"/>
      <xsd:choice minOccurs="0">
        <xsd:element ref="sld:RemoteOWS"/>
        <xsd:element ref="sld:InlineFeature"/>
      </xsd:choice>
      <xsd:choice minOccurs="0">
        <xsd:element ref="sld:LayerFeatureConstraints"/>
        <xsd:element ref="sld:LayerCoverageConstraints"/>
      </xsd:choice>
      <xsd:element ref="sld:UserStyle" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

```

Since a layer is defined as a collection of potentially mixed-type features, the **UserLayer** element must provide the means to identify the features to be used. All features to be rendered are assumed to be fetched from a Web Feature Server (WFS) or a Web Coverage Service (WCS, in which case the term “features” is used loosely). Alternatively they can be supplied in-line in the SLD document. This alternative is only recommended for small numbers of features of transient nature.

The remote server to be used is identified by **RemoteOWS** (OGC Web Service) element which is defined as follows:

```

<xsd:element name="RemoteOWS">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="sld:Service"/>
      <xsd:element ref="se:OnlineResource"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:element name="Service">

```

```

<xsd:simpleType>
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="WFS"/>
    <xsd:enumeration value="WCS"/>
  </xsd:restriction>
</xsd:simpleType>
</xsd:element>

```

It is hoped that this definition can be cleaned up to refer to schemas outside of the SLD definition. As indicated in Subclause 6.3, there are three ways to specify the WFS to be used: it may be identified by a **WFS** CGI parameter in a **GetMap** request; it may be given explicitly with the optional **RemoteOWS** element of the **UserLayer** element; or it may be the ‘default’ WFS for a WMS, which may be an implicit WFS built into the WMS.

The **InlineFeature** element is defined as follows:

```

<xsd:element name="InlineFeature">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="gml: FeatureCollection" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

```

A WCS is used similarly to a WFS in SLD. The WFS and WCS mechanisms in SLD may change as service chaining in OGC Web Services becomes more formalized.

The **DescribeFeatureType** and WFS **GetCapabilities** mechanisms may be used to interrogate the WMS or WFS for what feature types are available to be referenced, also as discussed in Subclause 6.3.

### 11.3.1 Feature Constraints

The **LayerFeatureConstraints** element is used to specify what features of what feature types are to be included in a layer. It is defined as:

```

<xsd:element name="LayerFeatureConstraints">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="sld:FeatureTypeConstraint" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:element name="FeatureTypeConstraint">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="se:FeatureTypeName" minOccurs="0"/>
      <xsd:element ref="ogc:Filter" minOccurs="0"/>
      <xsd:element ref="sld:Extent" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

```

When used in a **UserLayer**, the Extent reference defines what features are to be included in the layer and when used in a **NamedLayer**, it filters the features that are part of the named layer.

A **FeatureTypeConstraint** element is used to identify a feature type by a well-known name, using the **FeatureTypeName** element. Any positive number of **FeatureTypeConstraints** may be used to define the features of a layer, though all **FeatureTypeConstraints** in a **UserLayer** must originate from the same WFS source.

Named styles cannot be used with user-defined layers, since there is no general way to know if a named style is suitable for use with an arbitrary user-defined layer. Only user-defined styles may be used with user-defined layers, and any positive number of them may be used. Using zero styles is not allowed since there will be no pre-defined default style for an arbitrarily constructed layer.

Here is a simple example of an SLD that uses a user-defined layer (XML-namespaces definitions are omitted for brevity):

```
<StyledLayerDescriptor version="1.1.0">
  <UserLayer>
    <Name>MyLayer</Name>
    <RemoteOWS>
      <Service>WFS</Service>
      <OnlineResource xlink:type="simple" xlink:href="http://some.site.com/WFS?"/>
    </RemoteOWS>
    <LayerFeatureConstraints>
      <FeatureTypeConstraint>
        <FeatureTypeName>RoadFeatures</FeatureTypeName>
      </FeatureTypeConstraint>
    </LayerFeatureConstraints>
    <UserStyle>
      [...]
    </UserStyle>
  </UserLayer>
</StyledLayerDescriptor>
```

The WFS is named explicitly in the **UserLayer**, only a single feature type is included in the layer, and the **UserStyle** element is incomplete.

### 11.3.2 Coverage Constraints

The WCS **DescribeCoverage** and **GetCapabilities** mechanisms may be used to interrogate the WMS or WCS for what coverage data is available to be referenced, also as discussed in Section 6.3.

The **LayerCoverageConstraints** element is used to specify what subsets of what coverage offering are to be included in a layer. It is defined as:

```
<xsd:element name="LayerCoverageConstraints">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="sld:CoverageConstraint" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```



```

    </xsd:complexType>
  </xsd:element>

  <xsd:element name="CoverageConstraint">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="se:CoverageName"/>
        <xsd:element ref="sld:CoverageExtent" minOccurs="0"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="CoverageExtent">
    <xsd:annotation>
      <xsd:documentation>
        The CoverageExtent describes the time or range selections.
      </xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
      <xsd:choice>
        <xsd:element ref="sld:RangeAxis" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="sld:TimePeriod" minOccurs="0"/>
      </xsd:choice>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="RangeAxis">
    <xsd:annotation>
      <xsd:documentation>
        A RangeAxis describes the range selection for a coverage.
      </xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="se:Name"/>
        <xsd:element ref="sld:Value"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="Value" type="xsd:string"/>
  <xsd:element name="TimePeriod" type="xsd:string"/>

```

When used in a **UserLayer**, the **CoverageExtent** reference defines what coverage data is to be included in the layer and when used in a **NamedLayer**, it selects the data that are part of the named layer.

A **CoverageConstraint** element is used to identify a coverage offering by a well-known name, using the **CoverageName** element. Any positive number of **CoverageConstraints** may be used to define the coverage data of a layer, though all **CoverageConstraints** in a **UserLayer** must come from the same WCS source.

**TimePeriod** describes a subset corresponding to the specified time instants or intervals, expressed in an extended ISO 8601 syntax.

**RangeAxis** describes a range subset defined by a constraining parameter. The name of that parameter matches the name of an **AxisDescription** element in the range set

description of the selected coverage offering. The value is one of the acceptable values defined in the corresponding `AxisDescription` element.

Complex **TimePeriod** and **RangeAxis** values can be comma-separated list of intervals (2003-07-08T16:45:00Z/2003-07-08T20:30:00Z,2003-07-09T16:45:00Z or band1,band2,band3).

Named styles cannot be used with user-defined layers, since there is no general way to know if a named style is suitable for use with an arbitrary user-defined layer. Only user-defined styles may be used with user-defined layers, and any positive number of them may be used. Using zero styles is not allowed since there will be no pre-defined default style for an arbitrarily constructed layer.

Here is a simple example of an SLD that uses a user-defined layer (XML-namespaces definitions are omitted for brevity):

```
<StyledLayerDescriptor version="1.1.0">
  <UserLayer>
    <se:Name>MyLayer</se:Name>
    <RemoteOWS>
      <Service>WCS</Service>
      <se:OnlineResource xlink:type="simple" xlink:href="http://some.site.com/WCS?"/>
    </RemoteOWS>
    <LayerCoverageConstraints>
      <CoverageConstraint>
        <se:CoverageName>MOD_Grid_L2g_2d</se:CoverageName>
        <CoverageExtent>
          <RangeAxis>
            <se:Name>Band</se:Name>
            <Value>band1</Value>
          </RangeAxis>
        </CoverageExtent>
      </CoverageConstraint>
    </LayerCoverageConstraints>
    <UserStyle>
      [...]
    </UserStyle>
  </UserLayer>
</StyledLayerDescriptor>
```

This example shows the selection of a coverage data subset on a WCS serving ‘MOD09GHK’ data. The ‘MOD\_Grid\_L2g\_2d’ coverage offering has been selected.

This coverage has a range axis named ‘Band’ that has 3 discrete values which are the 3 channels of the coverage. This sample selects the first channel named ‘band1’.

It is the intersection of these criteria (name, range axis) that will determine the data that will be selected on the source coverage.

The WCS is named explicitly in the **UserLayer**, only a single coverage offering is included in the layer, and the **UserStyle** element is incomplete.

## 11.4 User-defined styles

A user-defined style allows map styling to be defined externally from a system and to be passed around in an interoperable format. The XML-Schema fragment for the **UserStyle** SLD element is as follows:

```
<xsd:element name="UserStyle">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="se:Name" minOccurs="0"/>
      <xsd:element ref="se:Description" minOccurs="0"/>
      <xsd:element ref="sld:IsDefault" minOccurs="0"/>
      <xsd:choice maxOccurs="unbounded">
        <xsd:element ref="se:FeatureTypeStyle"/>
        <xsd:element ref="se:CoverageStyle"/>
        <xsd:element ref="se:OnlineResource"/>
      </xsd:choice>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element name="IsDefault" type="xsd:boolean"/>
```

A **UserStyle** is at the same semantic level as a **NamedStyle** used in the context of a WMS. In a sense, a named style can be thought of as a reference to a hidden **UserStyle** that is stored inside of a map server.

The **Name** and **Description** elements allow a user-defined style to be identified and are optional. The given **Name** is equivalent to the name of a WMS named style and is used to reference the style externally when an SLD is used in ‘library mode’ (Subclause 9.2.2. The **Title** of the **Description** is a human-readable short description for the style that might be displayed in a GUI pick list, and the **Abstract** of the **Description** is a more exact description that may be a few paragraphs long.

The **IsDefault** element identifies whether a style is the default style of a layer, for use in SLD ‘library mode’ when rendering or for storing inside of a map server. **IsDefault** uses “1” or “true” for true and “0” or “false” for false. The default value is “0”.

A **UserStyle** can contain one or more **FeatureTypeStyles** or **CoverageStyles** which allow the rendering of features of specific types. These are described in Symbology Encoding. These styles can either be provided inline or they can be referenced using an **OnlineResource** containing an SE document with a **FeatureTypeStyle** or **CoverageStyle** root element. This organization allows the more convenient use of feature-style libraries.

Note that there is no restriction against a single **UserStyle** from including multiple **FeatureTypeStyles** that reference the same **FeatureTypeName**. This case does not create an exception in the rendering semantics, however, since a map styler is expected to process all **FeatureTypeStyles** in the order that they appear, regardless, plotting one instance over top of another.

The following is an (incomplete) example of a **UserStyle** used with a **NamedLayer**:

```
<StyledLayerDescriptor version="1.1.0">
  <NamedLayer>
    <se:Name>Transportation</se:Name>
    <UserStyle>
      <se:Description>
        <se:Name>GS1</se:Name>
        <se:Title>GeoSym</se:Title>
      </se:Description>
      <se:Abstract>GeoSym style for transportation</se:Abstract>
      <se:FeatureTypeStyle>
        [...]
      </se:FeatureTypeStyle>
    </UserStyle>
  </NamedLayer>
</StyledLayerDescriptor>
```

## **Annex A**

### **(normative)**

## **Abstract test suite**

### **A.1 General**

An SLD-enabled WMS has to conform to all conformance classes of the base WMS specification. In the following some general guidelines are given regarding the different classes of SLD-enabled WMS servers

### **A.2 Conformance Class A: Integrated SLD-WMS**

An integrated SLD-WMS has to support UserStyles, the DescribeLayer operation and either GetCapabilities (WFS) and DescribeFeatureType or GetCapabilities (WCS) and DescribeCoverage operation.

### **A.3 Conformance Class B. Component SLD-WMS**

All component SLD-WMS servers, regardless if they are of the FPS or CPS type have to support UserLayers and UserStyles.

### **A.3 Conformance Class C. Feature Portrayal Service**

A Feature Portrayal Service has to conform to Class B. Additionally it has to support RemoteWFS.

### **A.4 Conformance Class D. Coverage Portrayal Service**

A Coverage Portrayal Service has to conform to Class B. Additionally it has to support RemoteWCS.

**Annex B**  
(normative)

**XML schemas**

In addition to this document, this specification includes several normative XML Schema files. These are posted online at the URL <http://schemas.opengespatial.net/sld/1.1.0> where a lower level directory is used for this Version 1.1.0. These XML Schema files are also bundled in a zip file with the present document. In the event of a discrepancy between the bundled and online versions of the XML Schema files, the online files shall be considered authoritative.

The abilities now specified in this document use specified XML Schemas included in the zip file with this document. These XML Schemas combine the XML Schema fragments listed in various subclauses of this document, eliminating duplications. These XML Schema files are named:

StyledLayerDescriptor.xsd

All these XML Schemas 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 05-008].

## Annex C (informative)

### Example XML documents

#### D.1 Introduction

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

#### D.2 Capabilities document

```
<?xml version="1.0" encoding="UTF-8"?>
<wms:WMS_Capabilities version="1.3.0" xmlns="http://www.opengis.net/sld"
xmlns:wms="http://www.opengis.net/wms" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.opengis.net/sld
./sld_capabilities.xsd">
  <!-- Service Metadata -->
  <wms:Service>
    <!-- The WMT-defined name for this type of service -->
    <wms:Name>WMS</wms:Name>
    <!-- Human-readable title for pick lists -->
    <wms:Title>Acme Corp. Map Server</wms:Title>
    <!-- Narrative description providing additional information -->
    <wms:Abstract>Map Server maintained by Acme Corporation. Contact:
webmaster@wmt.acme.com. High-quality maps showing roadrunner nests and possible ambush
locations.</wms:Abstract>
    <wms:KeywordList>
      <wms:Keyword>bird</wms:Keyword>
      <wms:Keyword>roadrunner</wms:Keyword>
      <wms:Keyword>ambush</wms:Keyword>
    </wms:KeywordList>
    <!-- Top-level web address of service or service provider. See also OnlineResource
elements under <wms:DCPType>. -->
    <wms:OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="simple"
xlink:href="http://hostname/">
    <!-- Contact information -->
    <wms:ContactInformation>
      <wms:ContactPersonPrimary>
        <wms:ContactPerson>Jeff Smith</wms:ContactPerson>
        <wms:ContactOrganization>NASA</wms:ContactOrganization>
      </wms:ContactPersonPrimary>
      <wms:ContactPosition>Computer Scientist</wms:ContactPosition>
      <wms:ContactAddress>
        <wms:AddressType>postal</wms:AddressType>
        <wms:Address>NASA Goddard Space Flight Center</wms:Address>
        <wms:City>Greenbelt</wms:City>
        <wms:StateOrProvince>MD</wms:StateOrProvince>
        <wms:PostCode>20771</wms:PostCode>
        <wms:Country>USA</wms:Country>
      </wms:ContactAddress>
      <wms:ContactVoiceTelephone>+1 301 555-1212</wms:ContactVoiceTelephone>
```

```

    <wms:ContactElectronicMailAddress>user@host.com</wms:ContactElectronicMailAddress>
  </wms:ContactInformation>
  <!-- Fees or access constraints imposed. -->
  <wms:Fees>none</wms:Fees>
  <wms:AccessConstraints>none</wms:AccessConstraints>
  <wms:LayerLimit>16</wms:LayerLimit>
  <wms:MaxWidth>2048</wms:MaxWidth>
  <wms:MaxHeight>2048</wms:MaxHeight>
</wms:Service>
<wms:Capability>
  <wms:Request>
    <wms:GetCapabilities>
      <wms:Format>text/xml</wms:Format>
      <wms:DCPType>
        <wms:HTTP>
          <wms:Get>
            <wms:OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:type="simple" xlink:href="http://hostname/path?"/>
          </wms:Get>
          <wms:Post>
            <wms:OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:type="simple" xlink:href="http://hostname/path?"/>
          </wms:Post>
        </wms:HTTP>
      </wms:DCPType>
    </wms:GetCapabilities>
    <wms:GetMap>
      <wms:Format>image/gif</wms:Format>
      <wms:Format>image/png</wms:Format>
      <wms:Format>image/jpeg</wms:Format>
      <wms:DCPType>
        <wms:HTTP>
          <wms:Get>
            <!-- The URL here for invoking GetCapabilities using HTTP GET
is only a prefix to which a query string is appended. -->
            <wms:OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:type="simple" xlink:href="http://hostname/path?"/>
          </wms:Get>
        </wms:HTTP>
      </wms:DCPType>
    </wms:GetMap>
    <wms:GetFeatureInfo>
      <wms:Format>text/xml</wms:Format>
      <wms:Format>text/plain</wms:Format>
      <wms:Format>text/html</wms:Format>
      <wms:DCPType>
        <wms:HTTP>
          <wms:Get>
            <wms:OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:type="simple" xlink:href="http://hostname/path?"/>
          </wms:Get>
        </wms:HTTP>
      </wms:DCPType>
    </wms:GetFeatureInfo>
    <DescribeLayer>
      <wms:Format>text/xml</wms:Format>
      <wms:Format>text/plain</wms:Format>
      <wms:Format>text/html</wms:Format>
      <wms:DCPType>
        <wms:HTTP>
          <wms:Get>
            <wms:OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:type="simple" xlink:href="http://hostname/path?"/>
          </wms:Get>
        </wms:HTTP>
      </wms:DCPType>
    </DescribeLayer>
  </wms:Request>
</wms:Capability>
</wms:Service>

```



```

        </wms:Get>
      </wms:HTTP>
    </wms:DCPType>
  </DescribeLayer>
</wms:Request>
<wms:Exception>
  <wms:Format>XML</wms:Format>
  <wms:Format>INIMAGE</wms:Format>
  <wms:Format>BLANK</wms:Format>
</wms:Exception>
<UserDefinedSymbolization SupportSLD="1" UserLayer="1" UserStyle="1" RemoteWFS="1"
InlineFeature="1"/>
  <wms:Layer>
    <wms:Title>Acme Corp. Map Server</wms:Title>
    <wms:CRS>CRS:84</wms:CRS>
    <!-- all layers are available in at least this CRS -->
    <wms:AuthorityURL name="DIF_ID">
      <wms:OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="simple"
xlink:href="http://gcmd.gsfc.nasa.gov/difguide/whatisadif.html"/>
    </wms:AuthorityURL>
    <wms:Layer>
      <!-- This parent layer has a Name and can therefore be requested from a Map Server,
yielding a map of all subsidiary layers. -->
      <wms:Name>ROADS_RIVERS</wms:Name>
      <wms:Title>Roads and Rivers</wms:Title>
      <!-- See the spec to learn how some characteristics are inherited by
subsidiary layers. -->
      <wms:CRS>EPSG:26986</wms:CRS>
      <!-- An additional CRS for this layer -->
      <wms:EX_GeographicBoundingBox>
        <wms:westBoundLongitude>-71.63</wms:westBoundLongitude>
        <wms:eastBoundLongitude>-70.78</wms:eastBoundLongitude>
        <wms:southBoundLatitude>41.75</wms:southBoundLatitude>
        <wms:northBoundLatitude>42.90</wms:northBoundLatitude>
      </wms:EX_GeographicBoundingBox>
      <!-- The optional resx and resy attributes indicate the X and Y spatial
resolution in the units of that CRS. -->
      <wms:BoundingBox CRS="CRS:84" minx="-71.63" miny="41.75" maxx="-70.78"
maxy="42.90" resx="0.01" resy="0.01"/>
      <wms:BoundingBox CRS="EPSG:26986" minx="189000" miny="834000" maxx="285000"
maxy="962000" resx="1" resy="1"/>
      <!-- Optional Title, URL and logo image of data provider. -->
      <wms:Attribution>
        <wms:Title>State College University</wms:Title>
        <wms:OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="simple"
xlink:href="http://www.university.edu/">
          <wms:LogoURL width="100" height="100">
            <wms:Format>image/gif</wms:Format>
            <wms:OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="simple"
xlink:href="http://www.university.edu/icons/logo.gif"/>
          </wms:LogoURL>
        </wms:Attribution>
        <!-- Identifier whose meaning is defined in an AuthorityURL element -->
        <wms:Identifier authority="DIF_ID">123456</wms:Identifier>
        <wms:FeatureListURL>
          <wms:Format>XML</wms:Format>
          <wms:OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="simple"
xlink:href="http://www.university.edu/data/roads_rivers.gml"/>
        </wms:FeatureListURL>
        <wms:Style>
          <wms:Name>USGS</wms:Name>
          <wms:Title>USGS Topo Map Style</wms:Title>

```

```

        <wms:Abstract>Features are shown in a style like that used in USGS topographic
maps.</wms:Abstract>
        <!-- A picture of a legend for a Layer in this Style -->
        <wms:LegendURL width="72" height="72">
          <wms:Format>image/gif</wms:Format>
          <wms:OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="simple"
xlink:href="http://www.university.edu/legends/usgs.gif"/>
        </wms:LegendURL>
        <!-- An XSL stylesheet describing how feature data will rendered to create
a map of this layer. -->
        <wms:StyleSheetURL>
          <wms:Format>text/xsl</wms:Format>
          <wms:OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="simple"
xlink:href="http://www.university.edu/stylesheet/usgs.xml"/>
        </wms:StyleSheetURL>
      </wms:Style>
    <wms:Layer queryable="1">
      <wms:Name>ROADS_1M</wms:Name>
      <wms:Title>Roads at 1:1M scale</wms:Title>
      <wms:Abstract>Roads at a scale of 1 to 1 million.</wms:Abstract>
      <wms:KeywordList>
        <wms:Keyword>road</wms:Keyword>
        <wms:Keyword>transportation</wms:Keyword>
        <wms:Keyword>atlas</wms:Keyword>
      </wms:KeywordList>
      <wms:Identifier authority="DIF_ID">123456</wms:Identifier>
      <wms:MetadataURL type="FGDC:1998">
        <wms:Format>text/plain</wms:Format>
        <wms:OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="simple"
xlink:href="http://www.university.edu/metadata/roads.txt"/>
      </wms:MetadataURL>
      <wms:MetadataURL type="ISO19115:2003">
        <wms:Format>text/xml</wms:Format>
        <wms:OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="simple"
xlink:href="http://www.university.edu/metadata/roads.xml"/>
      </wms:MetadataURL>
      <!-- In addition to the Style specified in the parent Layer, this Layer is
available in this style. -->
      <wms:Style>
        <wms:Name>ATLAS</wms:Name>
        <wms:Title>Road atlas style</wms:Title>
        <wms:Abstract>Roads are shown in a style like that used in a commercial road
atlas.</wms:Abstract>
        <wms:LegendURL width="72" height="72">
          <wms:Format>image/gif</wms:Format>
          <wms:OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:type="simple" xlink:href="http://www.university.edu/legends/atlas.gif"/>
        </wms:LegendURL>
      </wms:Style>
    </wms:Layer>
    <wms:Layer queryable="1">
      <wms:Name>RIVERS_1M</wms:Name>
      <wms:Title>Rivers at 1:1M scale</wms:Title>
      <wms:Abstract>Rivers at a scale of 1 to 1 million.</wms:Abstract>
      <wms:KeywordList>
        <wms:Keyword>river</wms:Keyword>
        <wms:Keyword>canal</wms:Keyword>
        <wms:Keyword>waterway</wms:Keyword>
      </wms:KeywordList>
    </wms:Layer>
  </wms:Layer>
  <wms:Layer queryable="1">
    <wms:Title>Weather Forecast Data</wms:Title>

```

```

<wms:CRS>CRS:84</wms:CRS>
<!-- harmless repetition of common CRS -->
<wms:EX_GeographicBoundingBox>
  <wms:westBoundLongitude>-180</wms:westBoundLongitude>
  <wms:eastBoundLongitude>180</wms:eastBoundLongitude>
  <wms:southBoundLatitude>-90</wms:southBoundLatitude>
  <wms:northBoundLatitude>90</wms:northBoundLatitude>
</wms:EX_GeographicBoundingBox>
<!-- These weather data are available daily from 1999-01-01 through
2000-08-22. -->
  <wms:Dimension name="time" units="ISO8601" default="2000-08-22">
1999-01-01/2000-08-22/P1D
</wms:Dimension>
  <wms:Layer>
    <wms:Name>Clouds</wms:Name>
    <wms:Title>Forecast cloud cover</wms:Title>
  </wms:Layer>
  <wms:Layer>
    <wms:Name>Temperature</wms:Name>
    <wms:Title>Forecast temperature</wms:Title>
  </wms:Layer>
  <wms:Layer>
    <wms:Name>Pressure</wms:Name>
    <wms:Title>Forecast barometric pressure</wms:Title>
    <!-- This Pressure layer is available at several elevations and times. -->
    <wms:Dimension name="elevation" units="EPSG:5030"/>
    <wms:Dimension name="time" units="ISO8601" default="2000-08-22">
1999-01-01/2000-08-22/P1D</wms:Dimension>
    <wms:Dimension name="elevation" units="CRS:88" default="0" nearestValue="1">
0,1000,3000,5000,10000</wms:Dimension>
  </wms:Layer>
</wms:Layer>
<!-- Example of a layer which is a static map of fixed
size which the server cannot subset or make transparent -->
  <wms:Layer opaque="1" noSubsets="1" fixedWidth="512" fixedHeight="256">
    <wms:Name>ozone_image</wms:Name>
    <wms:Title>Global ozone distribution (1992)</wms:Title>
    <wms:EX_GeographicBoundingBox>
      <wms:westBoundLongitude>-180</wms:westBoundLongitude>
      <wms:eastBoundLongitude>180</wms:eastBoundLongitude>
      <wms:southBoundLatitude>-90</wms:southBoundLatitude>
      <wms:northBoundLatitude>90</wms:northBoundLatitude>
    </wms:EX_GeographicBoundingBox>
    <wms:Dimension name="time" units="ISO8601" default="1992">1992</wms:Dimension>
  </wms:Layer>
<!-- Example of a layer which originated from another WMS and has been
"cascaed" by this WMS. -->
  <wms:Layer cascaded="1">
    <wms:Name>population</wms:Name>
    <wms:Title>World population, annual</wms:Title>
    <wms:EX_GeographicBoundingBox>
      <wms:westBoundLongitude>-180</wms:westBoundLongitude>
      <wms:eastBoundLongitude>180</wms:eastBoundLongitude>
      <wms:southBoundLatitude>-90</wms:southBoundLatitude>
      <wms:northBoundLatitude>90</wms:northBoundLatitude>
    </wms:EX_GeographicBoundingBox>
    <wms:Dimension name="time" units="ISO8601"
default="2000">1990/2000/P1Y</wms:Dimension>
  </wms:Layer>
</wms:Layer>
</wms:Capability>
</wms:WMS_Capabilities>

```

## D.2 SLD document

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<StyledLayerDescriptor version="1.1.0" xsi:schemaLocation="http://www.opengis.net/sld
StyledLayerDescriptor.xsd" xmlns="http://www.opengis.net/sld" xmlns:ogc="http://www.opengis.net/ogc"
xmlns:se="http://www.opengis.net/se" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <NamedLayer>
    <se:Name>OCEANSEA_1M:Foundation</se:Name>
    <UserStyle>
      <se:Name>GEOSYM</se:Name>
      <IsDefault>1</IsDefault>
      <se:FeatureTypeStyle>
        <se:FeatureTypeName>Foundation</se:FeatureTypeName>
        <se:Rule>
          <se:Name>main</se:Name>
          <se:PolygonSymbolizer uom="http://www.opengeospatial.org/sld/units/pixel">
            <se:Name>MySymbol</se:Name>
            <se:Description>
              <se:Title>Example Symbol</se:Title>
              <se:Abstract>This is just a simple example.</se:Abstract>
            </se:Description>
            <se:Geometry>
              <ogc:PropertyName>GEOMETRY</ogc:PropertyName>
            </se:Geometry>
            <se:Fill>
              <se:SvgParameter name="fill">#96C3F5</se:SvgParameter>
            </se:Fill>
          </se:PolygonSymbolizer>
        </se:Rule>
      </se:FeatureTypeStyle>
    </UserStyle>
  </NamedLayer>
</StyledLayerDescriptor>

```

## D.3 DescribeLayer response

```

<?xml version="1.0" encoding="UTF-8"?>
<DescribeLayerResponse xmlns="http://www.opengis.net/sld" xmlns:ows="http://www.opengis.net/ows"
xmlns:se="http://www.opengis.net/se" xmlns:wfs="http://schemas.opengis.net/wfs"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:myns="http://myserver/somenamespace"
xsi:schemaLocation="http://www.opengis.net/sld DescribeLayer.xsd">
  <Version>1.1.0</Version>
  <LayerDescription>
    <ows:Type>wfs</ows:Type>
    <se:OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="simple"
xlink:href="http://mywfs/server/ogcwebservice"/>
    <TypeName>
      <se:FeatureTypeName>myns:cloudfeatures</se:FeatureTypeName>
    </TypeName>
  </LayerDescription>
</DescribeLayerResponse>

```

## D.4 XML-encoded GetMap request

```

<?xml version="1.0" encoding="UTF-8"?>
<GetMap xmlns="http://www.opengis.net/sld" xmlns:gml="http://www.opengis.net/gml"
xmlns:ogc="http://www.opengis.net/ogc" xmlns:ows="http://www.opengis.net/ows"
xmlns:se="http://www.opengis.net/se" xmlns:wms="http://www.opengis.net/wms"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.opengis.net/sld
GetMap.xsd" version="1.3.0">
  <StyledLayerDescriptor version="1.1.0">
    <NamedLayer>
      <se:Name>Rivers</se:Name>
      <NamedStyle>
        <se:Name>CenterLine</se:Name>
      </NamedStyle>
    </NamedLayer>
    <NamedLayer>
      <se:Name>Roads</se:Name>
      <NamedStyle>
        <se:Name>CenterLine</se:Name>
      </NamedStyle>
    </NamedLayer>
    <NamedLayer>
      <se:Name>Houses</se:Name>
      <NamedStyle>
        <se:Name>Outline</se:Name>
      </NamedStyle>
    </NamedLayer>
  </StyledLayerDescriptor>
  <CRS>EPSG:4326</CRS>
  <BoundingBox crs="http://www.opengis.net/gml/srs/epsg.xml#4326">
    <ows:LowerCorner>-180.0 -90.0</ows:LowerCorner>
    <ows:UpperCorner>180.0 90.0</ows:UpperCorner>
  </BoundingBox>
  <Output>
    <Size>
      <Width>1024</Width>
      <Height>512</Height>
    </Size>
    <wms:Format>image/jpeg</wms:Format>
    <Transparent>>false</Transparent>
  </Output>
  <Exceptions>XML</Exceptions>
</GetMap>

```