Open Geospatial Consortium Inc.

Date: 2005-10-19

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

Version: 1.1.0 (draft)

Category: OpenGIS® Discussion Paper

Editors: Dr. Markus Müller, James MacGill

Styled Layer Descriptor Application Profile of the Web Map Service: Draft Implementation Specification

Warning

This proposed document is not an OGC Standard. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an OGC Standard.

Document type:	OpenGIS® Discussion Paper
Document subtype:	Application Profile
Document stage:	Approved
Document language:	English

Contents

Page

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 ... 3
- 6.1 A review of WMS 1.3 ... 3
- 6.2 General HTTP request rules as used by WMS and SLD ... 5
- 6.3 Styled-Layer Descriptor ... 5
- 6.4 WMS requests using an SLD ... 6
- 6.5 `GetMap` POST method ... 8
- 6.6 Web Map Servers and Web Feature/Coverage Servers ... 9

7 DescribeLayer operation (optional) ... 12
- 7.1 Introduction ... 12
- 7.2 DescribeLayer operation request (TBD) ... 13
  - 7.2.1 DescribeLayer request parameters ... 13
  - 7.2.2 TBD request KVP encoding (optional or required) ... 13
- 7.3 TBD operation response ... 14
  - 7.3.1 Normal response parameters ... 14
  - 7.3.2 Normal response XML encoding ... 15
  - 7.3.3 TBD exceptions ... 17
- 7.4 Examples ... 17

8 GetLegendGraphic operation (optional) ... 18
- 8.1 Introduction ... 18
- 8.2 TBD operation request ... 21
  - 8.2.1 TBD request parameters ... 21
  - 8.2.2 TBD request KVP encoding (optional or required) ... 22
  - 8.2.3 TBD request XML encoding (required or optional) ... 22
- 8.3 TBD operation response ... 27
  - 8.3.1 Normal response parameters ... 27
  - 8.3.2 Normal response XML encoding ... 28
  - 8.3.3 TBD exceptions ... 29
- 8.4 Examples ... 30

9 SLD encoding ... 31
- 9.1 SLD root element ... 31
- 9.2 Named layers ... 32

9.3 User-defined layers....35
9.3.1 Feature Constraints 36
9.3.2 Coverage Constraints....38
9.4 User-defined styles....40

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. The WMS 1.3 Specification, OGC 06-042, is the most recent OGC member approved version of that 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 documents 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 (Co-Editor)
Pennsylvania State University. (Co-Editor)
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	Penn State
Dimitri Monie	Ionic Software s.a.
Dr. Markus Müller	lat/lon GmbH
Dr. Andreas Poth	lat/lon GmbH
Raj Singh	Syncline
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	Fith 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		Markus Müller	All	Split SLD specification in SLD profile for WMS (this document) and Symbology Encoding
2006-20-04		C. Reed	Various	Get ready for posting as a DP

vi. Changes to the OGC Abstract Specification

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

Foreword

This document together with OGC 05-077 (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.

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 NGA'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) Implementation 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 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 7 and 8) 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 **`FeatureStyle`** 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 OpenGIS® 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>.

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>.

OGC Adopted Implementation Specification: Filter Encoding version 1.1, May 2004, OGC document OGC 04-095 <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>.

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>.

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

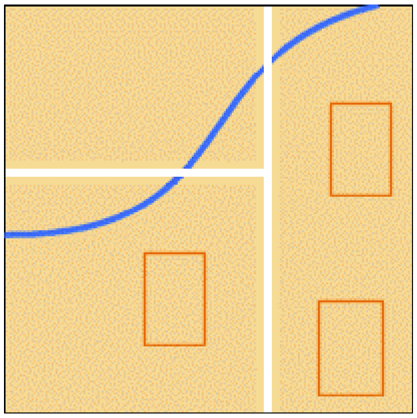
WMS 1.3 and earlier versions describe the appearance of a map in terms of 'styled layers'. A styled layer can be considered as a transparent sheet with features symbolized upon it. A map is made up of a number of these styled layers put together in a specified order. The styled layers are said to be 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:

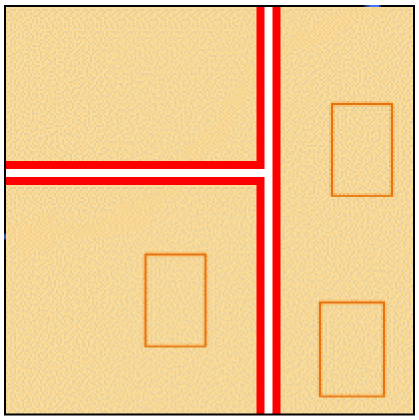
```
a) Houses:Outline
b) Roads:CenterLine
c) Rivers:CenterLine
```

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 over 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 General HTTP request rules as used by WMS and SLD

HTTP supports two request methods: GET and POST. One or both of these methods may be defined for a particular OGC Web Service type and offered by a service instance, and the use of the Online Resource URL differs in each case. The basic WMS specification defines HTTP GET (mandatory) and HTTP POST (optional) for invoking operations.

6.3 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 Clauses TBD. 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 an example 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 and the "painter's model" is also used for Z-ordering. An SLD XML document can become much more complex with user-defined styling.

6.4 WMS requests using an SLD

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 and including an embedded SLD, as described in section 6.3.4 of WMS 1.3 specification.

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.

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 be able to place the SLD document in a Web location accessible to the WMS.

Consider the example (incomplete) `GetMap` request from the previous 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 **Error! Reference source not found.**, 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 TBD, 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.

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.

6.5 `GetMap` POST method

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

TBD: The WMS **`GetMap`** schema is presented in Annex X.X and the SLD **`GetMap`** schema is presented in Annex Y.Y.

Using the POST **`GetMap`** method, the running example translates into the following XML encoding [TBD: adjust to WMS 1.3 POST Schema - if there is one or eventually create one]:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE GetMap
  SYSTEM "http://some.site.com/wms/GetMap.xsd">
<ogc:GetMap xmlns:ogc="http://www.opengis.net/ows"
            xmlns:gml="http://www.opengis.net/gml"
            env:encodingStyle=
              "http://www.w3.org/2001/09/soap-encoding"
            version="1.3.0" service="WMS">
   <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>
   <BoundingBox
      srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
      <gml:coord>
         <gml:X>-180.0</gml:X>
         <gml:Y>-90.0</gml:Y>
      </gml:coord>
      <gml:coord>
         <gml:X>180.0</gml:X>
         <gml:Y>90.0</gml:Y>
      </gml:coord>
   </BoundingBox>
   <Output>
      <Format>image/jpeg</Format>
      <Transparent>false</Transparent>
      <Size>
```

```
          <Width>1024</Width>
          <Height>512</Height>
       </Size>
    </Output>
    <Exceptions>application/vnd.ogc.se+xml</Exceptions>
</ogc: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.

6.6 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. (Previously, this mechanism was provided with a single `WFS` parameter, but that was too restrictive.) 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&
   SRS=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.

- 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 the ‘two ends’ of the spectrum, represented by a ‘component’ WMS at one end and ‘integrated’ WMS at the other.

- Component WMS

  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 probably 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.

  It would be expected that WMS based upon XSLT technology would fit into this category.

- Integrated WMS

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

  - An integrated WMS probably has pre-defined ‘named’ layers and styles.
  - An integrated WMS supports the WMS interface and the **`DescribeFeatureType`** request of the WFS interface or the **`GetCapabilities`** and **`DescribeCoverage`** requests of the WCS interface.

- o An integrated WMS can only symbolize feature data from its own internal the feature store.
- o An integrated WMS might only support 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 and a minimal set of WFS/WCS operations. It must support the **`DescribeFeatureType`** request of a WFS or the **`GetCapabilities`** and **`DescribeCoverage`** of a WCS. This describes the properties of a feature/coverage type specified by name in the request. And, if the WMS supports user-defined layers, then it must support the WFS **`GetCapabilities`** request. For a WFS this returns, among other things, the names of all the feature types supported by the WFS. Together the two WFS requests allow clients to retrieve all the information they require to construct user-defined symbolizations. The WCS **`GetCapabilities`** and Describe Coverage requests allow clients to retrieve all the information they require to construct user-defined symbolizations.

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 it should be used (see Clause TBD); 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 DescribeLayer operation (optional)

7.1 Introduction

Defining a user-defined style requires information about the features being symbolized, or at least their feature 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 types. Note that it is perfectly valid for a named layer not to be describable in this way. It has been suggested that we reuse the WFS mechanism for indicating how one identifies feature types in a WFS, namely by using the **Query** element. Annex B gives the TBD XML Schema for the response.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE WMS_DescribeLayerResponse
  SYSTEM "http://some.site.com/sld/DSR.dtd" >
<WMS_DescribeLayerResponse version="1.3.0" >
   <!-- 'Layer_A' comes from the wfs specified by
   the prefix "http://www.mywfs.com/WFS?" and has features
   of types 'Road_FT' and 'Route_FT' -->
   <LayerDescription name="Layer_A"
                     wfs="http://www.mywfs.com/WFS?">
      <Query typeName="Road_FT" />
      <Query typeName="Route_FT" />
   </LayerDescription>
   <!-- 'Layer_B' cannot be described in terms of
   a WFS and so has no wfs attribute and no contents -->
   <LayerDescription name="Layer_B">
   </LayerDescription>
</WMS_DescribeLayerResponse>
```

Note that even an integrated WMS should provide a WFS/WCS prefix, since this allows **`DescribeFeatureType`** requests to be made.

A WMS need not support the **`DescribeLayer`** request. However, it should be noted that it may be common for a WMS not to support **`UserLayers`** but to allow **`UserStyles`** to be applied to named layers. In such circumstances, supporting the **`DescribeLayer`** request is the only interoperable way in which a client can specify user-defined symbolization.

7.2 DescribeLayer operation request (TBD)

7.2.1 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]. The next DescribeLayer parameters listed below are copied from Table 1 in Subclause 7.2 of this document.

Table 7 — Parameters in DescribeLayer operation request

Name [a]	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	Character String type, not empty Value is specified by each Implementation Specification and Schemas version	One (mandatory)
layers	names of layers description requestesd of	Character String type, not empty. Multiple layer names separated by ‘,’	Multiple (one is 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.

7.2.2 TBD request KVP encoding (optional or 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 8 — DescribeLayer operation request URL parameters

Name and example [a]	Optionality and use	Definition and format
service=WMS	Mandatory	Service type identifier
request= DescribeLayer	Mandatory	Operation name
version=1.3.0	Mandatory	Specification and schema version for this operation
layers=layer_list	Mandatory	names of layers that description is requested for
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].		

7.3 TBD operation response

7.3.1 Normal response parameters

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 types. Note that it is perfectly valid for a named layer not to be describable in this way. It has been suggested that we reuse the WFS mechanism for indicating how one identifies feature types in a WFS, namely by using the **`Query`** element. Annex B gives the DTD for the response.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE WMS_DescribeLayerResponse
  SYSTEM "http://some.site.com/sld/DSR.dtd" >
<WMS_DescribeLayerResponse version="1.1.0" >
   <!-- 'Layer_A' comes from the wfs specified by
   the prefix "http://www.mywfs.com/WFS?" and has features
   of types 'Road_FT' and 'Route_FT' -->
   <LayerDescription name="Layer_A"
                     wfs="http://www.mywfs.com/WFS?">
      <Query typeName="Road_FT" />
      <Query typeName="Route_FT" />
   </LayerDescription>
   <!-- 'Layer_B' cannot be described in terms of
   a WFS and so has no wfs attribute and no contents -->
   <LayerDescription name="Layer_B">
   </LayerDescription>
</WMS_DescribeLayerResponse>
```

Note that even an integrated WMS should provide a WFS/WCS prefix, since this allows **`DescribeFeatureType`** requests to be made.

A WMS need not support the **`DescribeLayer`** request. However, it should be noted that it may be common for a WMS not to support **`UserLayers`** but to allow **`UserStyles`** to be applied to named layers. In such circumstances, supporting the **`DescribeLayer`** request is the only interoperable way in which a client can specify user-defined symbolization.

The normal response to a valid TBD operation request shall be TBD. More precisely, a response from the TBD operation shall include the parts listed in Table 9. This table also specifies the UML model data type plus the multiplicity and use of each listed part.

Table 9 — Parts of TBD operation response

Name	Definition	Data type and use	Multiplicity and use
a			

NOTE The UML class diagram contained in Subclause C.TBD provides a graphical view of the contents of the TBD operation response listed in Tables 9- TBD.

7.3.2 Normal response XML encoding

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

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns:wcts="http://www.opengeospatial.net/wcts"
xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.opengeospatial.net/wcts"
elementFormDefault="qualified" xml:lang="en">
   <annotation>
      <appinfo>isTransformableResponse.xsd 2004/10/27</appinfo>
      <documentation>This XML Schema encodes the WCTS IsTransformable
operation message. </documentation>
   </annotation>
   <!-- =============================================================
      elements and types
   ============================================================= -->
   <element name="IsTransformableResponse">
      <annotation>
         <documentation>Response to a valid IsTransformable operation
request sent to a WCTS. </documentation>
      </annotation>
      <complexType>
         <sequence/>
         <attribute name="transformable" type="boolean"
use="required">
            <annotation>
               <documentation>Indicates whether this WCTS server can
perform a transformation from the sourceCRS to the targetCRS identified
in the operation request. The value shall be "true" or "false".
</documentation>
            </annotation>
```

```
          </attribute>
          <attribute name="problem" type="wcts:ProblemType"
use="optional">
             <annotation>
                <documentation>Type of transformation problem detected
by WCTS server. This attribute shall be included whenever the
"transformable" attribute is false. </documentation>
             </annotation>
          </attribute>
       </complexType>
    </element>
    <!-- ============================================================ -->
    <simpleType name="ProblemType">
       <annotation>
          <documentation>Type of transformation problem by WCTS server.
</documentation>
       </annotation>
       <restriction base="string">
          <enumeration value="source CRS">
             <annotation>
                <documentation>WCTS server cannot transform from
identified source CRS. </documentation>
             </annotation>
          </enumeration>
          <enumeration value="target CRS">
             <annotation>
                <documentation>WCTS server cannot transform to
identified target CRS from identified source CRS. </documentation>
             </annotation>
          </enumeration>
          <enumeration value="geometry type">
             <annotation>
                <documentation>WCTS server cannot transform one or more
identified geometry types. </documentation>
             </annotation>
          </enumeration>
          <enumeration value="coverage type">
             <annotation>
                <documentation>WCTS server cannot transform one or more
identified coverage types. </documentation>
             </annotation>
          </enumeration>
          <enumeration value="interpolation method">
             <annotation>
                <documentation>WCTS server cannot perform one or more
identified interpolation methods. </documentation>
             </annotation>
          </enumeration>
          <enumeration value="other">
             <annotation>
                <documentation>WCTS server cannot perform identified
transformation due to some other problem, including incompatibility
between identified parameters. </documentation>
             </annotation>
          </enumeration>
       </restriction>
    </simpleType>
</schema>
```

7.3.3 TBD exceptions

When a TBD server encounters an error while performing a TBD 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 TBD 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
TBD	TBD	TBD

7.4 Examples

A TBD operation request for TBD can look like this encoded in XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<IsTransformable xmlns="http://www.opengeospatial.net/wcts"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengeospatial.net/wcts
isTransformable.xsd" service="WCTS" version="0.0.0">
   <!-- Primary editor: Arliss Whiteside. Last updated 2003/12/02. -->
   <!-- This template can be used for all CRSs defined by the EPSG. -->
   <sourceCRS xlink:href="urn:ogc:srs:EPSG::4326"/>
   <targetCRS xlink:href="urn:ogc:srs:EPSG::23032"/>
   <geometryType>LineStringType</geometryType>
</IsTransformable>
```

An example response to a TBD operation request is:

```
<?xml version="1.0" encoding="UTF-8"?>
<IsTransformableResponse xmlns="http://www.opengeospatial.net/wcts"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengeospatial.net/wcts
isTransformableResponse.xsd" transformable="true"/>
<!-- Primary editor: Arliss Whiteside. Last updated 2003/12/02. -->
```

8 GetLegendGraphic operation (optional)

8.1 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 format.

The structuring of SLD **UserStyle**s into SE **FeatureStyles** 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:

Roads Layer	
	Highways
	Collector Roads
	Minor Roads

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 or 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 `LegendURL`s.

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` (which is the way that SLD is really intended to be used but nobody implements it), 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 GET parameters of the `GetLegendGraphic` operation are defined as follows:

Parameter	**Required**	**Description**
`VERSION`	Required	Version as required by OGC interfaces.

Parameter	**Required**	**Description**
`REQUEST`	Required	Value must be "**`GetLegendGraphic`**".
`LAYER`	Required	Layer for which to produce legend graphic.
`STYLE`	Optional	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.
`FEATURETYPE`	Optional	Feature type for which to produce the legend graphic. This is not needed if the layer has only a single feature type.
`RULE`	Optional	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.
`SCALE`	Optional	In the case that a **`RULE`** is not specified for a style, this parameter may assist the server in selecting a more appropriate representative graphic by eliminating internal rules that are out-of-scope. This value is a standardized scale denominator, defined in Subclause **Error! Reference source not found.**
`SLD`	Optional	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.
`SLD_BODY`	Optional	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.
`FORMAT`	Required	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.
`WIDTH`	Optional	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.
`HEIGHT`	Optional	This gives a hint for the height of the returned graphic in pixels.

Parameter	Required	Description
`EXCEPTIONS`	Optional	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.

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.

Here is an example invocation:

```
  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.

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 SLD `Rule` (defined in Subclause **Error! Reference source not found.**) 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 rendered in order to get a good representative image. So, this is what the `LegendGraphic` SLD 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
```

8.2 TBD operation request

8.2.1 TBD request parameters

A request to perform the TBD 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]. The next TBD parameters listed below are copied from Table 1 in Subclause 7.2 of this document.

In all tables such as the following, all cells should use the style "Body Text indent". In the top row only, the cell contents shall be bold and centered. In the remaining rows, cell contents shall not be bold and shall be left justified. I recommend not including periods in any column.

The left column should list the parameter name using the XML encoding capitalization specified in Subclause 11.6.2 of [OGC 05-008]. To break long names at appropriate places, insert at such places a special character using Insert/Symbol/Special Characters/No-Width Optional Break.

The second column should list the definition of this parameter, omitting un-necessary words such as "a", "the", and "is". If the parameter value is the identifier of something, not a description or definitions, say that this parameter is "Identifier of TBD".

In the third column, the first item should state the data type used for this parameter, using data types appropriate in a UML model, in which this parameter is a named attribute of a UML class. If this parameter is not simple, but is a data structure, name that data structure and refer to a separate table that shall be used to specify the contents of that data structure. The second item in the third column should indicate the source of values for this parameter, the alternative values, or other value information, unless the values are quite clear from other listed information.

In the right (fourth) column, the first item should be the multiplicity and optionality of this parameter in this data structure, either "One (mandatory)", "One or more (mandatory)", "Zero or one (optional)", or "Zero or more (optional)". (Yes, these are redundant, but I think ISO wants this information.) The second item in the right column should specify how any multiplicity other than "One (mandatory)" shall be used. If that parameter is optional, under what condition(s) shall that parameter be included or not included? If that parameter can be repeated, for what is that parameter repeated? (These conditions may seem obvious to you, but they are rarely obvious to most readers.)

Table 7 — Parameters in TBD operation request

Name [a]	**Definition**	**Data type and value**	**Multiplicity and use**
service	Service type identifier	Character String type, not empty	One (mandatory)

		Value is OWS type abbreviation (e.g., "WMS", "WFS")	
request	Operation name	Character String type, not empty Value is operation name (e.g., "GetCapabilities")	One (mandatory)
version	Specification version for operation	Character String type, not empty Value is specified by each Implementation Specification and Schemas version	One (mandatory)
TBD	TBD	TBD	TBD
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.

NOTE 3 The UML class diagram contained in Subclause TBD provides a useful graphical view of the contents of the TBD operation request listed in Tables TBD - TBD.

8.2.2 TBD request KVP encoding (optional or required)

Servers can implement HTTP GET transfer of the TBD operation request, using KVP encoding. The KVP encoding of the TBD 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 8 — TBD operation request URL parameters

Name and example [a]	Optionality and use	Definition and format
service=WCTS	Mandatory	Service type identifier
request= IsTransformable	Mandatory	Operation name
version=TBD	Mandatory	Specification and schema version for this operation
TBD=TBD	TBD	TBD
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].		

8.2.3 TBD request XML encoding (required or optional)

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

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<schema xmlns:wcts="http://www.opengeospatial.net/wcts"
xmlns:ows="http://www.opengeospatial.net/ows"
xmlns:gml="http://www.opengis.net/gml"
xmlns:xlink="http://opengeospatial.net/xlink"
xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.opengeospatial.net/wcts"
elementFormDefault="qualified" xml:lang="en">
   <annotation>
      <appinfo>fragmentIsTransformableRequest.xsd 2004/12/20</appinfo>
      <documentation>This XML Schema encodes the WCTS IsTransformable
operation request message. </documentation>
   </annotation>
   <!-- ==============================================================
      includes and imports
   ============================================================== -->
   <import namespace="http://www.opengeospatial.net/ows"
schemaLocation="owsAdditions.xsd"/>
   <import namespace="http://www.opengis.net/gml"
schemaLocation="../gml/3.1.1/base/coordinateReferenceSystems.xsd"/>
   <!-- ==============================================================
      elements and types
   ============================================================== -->
   <element name="IsTransformable">
      <annotation>
         <documentation>Request to a WCTS to perform the
IsTransformable operation. This operation allows clients to check if
transformation of a specific set of geometry and/or coverage types is
possible between two coordinate reference systems. Either the desired
source and target CRSs can be directly identified, or a specific
coordinate transformation between two CRSs can be identified. This
operation will check if the identified geometries are supported and if
there is a valid way (sequence of transformation steps) to transform
the coordinates from the source CRS to the target CRS. (This operation
will not check if this transformation makes any sense.) In this XML
encoding, no "request" parameter is included, since the element name
specifies the specific operation. </documentation>
      </annotation>
      <complexType>
         <sequence>
            <choice>
               <sequence>
                  <group ref="wcts:SourceAndTargetCRSs">
                     <annotation>
                        <documentation>Desired well-known SourceCRS
and TargetCRS, included when client is not specifying a specific
coordinate operation. </documentation>
                     </annotation>
                  </group>
               </sequence>
               <element name="Transformation" type="anyURI">
                  <annotation>
                     <documentation>Desired well-known coordinate
operation, included when client is specifying a specific coordinate
operation. </documentation>
                  </annotation>
               </element>
               <element name="Method" type="anyURI">
                  <annotation>
```

```
                        <documentation>Desired well-known operation
method that can be used in a user-defined coordinate operation,
included when client is considering specifying a specific user-defined
coordinate operation. </documentation>
                    </annotation>
                </element>
            </choice>
            <choice>
                <element name="GeometryType"
type="ows:GeometryTypeType" maxOccurs="unbounded">
                    <annotation>
                        <documentation>Unordered list of one or more GML
3 geometric primitive types that a client can request be transformed by
a WCTS server. It is assumed that a WCTS server can also transform the
corresponding geometric complexes and aggregates. </documentation>
                    </annotation>
                </element>
                <sequence>
                    <element name="CoverageType"
type="ows:CoverageTypeType" maxOccurs="unbounded">
                        <annotation>
                            <documentation>Unordered list of one or more
GML 3 coverage types that a client can request be transformed by a WCTS
server. </documentation>
                        </annotation>
                    </element>
                    <element name="InterpolationMethod"
type="ows:InterpolationMethodType" minOccurs="0" maxOccurs="unbounded">
                        <annotation>
                            <documentation>Unordered list of zero or more
interpolation methods that a client can request be performed on a
coverage by a WCTS server. An interpolation is used after coverage
points have been transformed. </documentation>
                        </annotation>
                    </element>
                    <element name="GeometryType"
type="ows:GeometryTypeType" minOccurs="0" maxOccurs="unbounded">
                        <annotation>
                            <documentation>Unordered list of zero or more
GML 3 geometric primitive types that a client can request be
transformed by a WCTS server. It is assumed that a WCTS server can also
transform the corresponding geometric complexes and aggregates.
</documentation>
                        </annotation>
                    </element>
                </sequence>
            </choice>
        </sequence>
        <attribute name="service" type="wcts:ServiceType"
use="required" fixed="WCTS"/>
        <attribute name="version" type="wcts:VersionType"
use="required"/>
    </complexType>
</element>
<!-- ========================================================== -->
<group name="SourceAndTargetCRSs">
    <annotation>
```

```
          <documentation>Group combining SourceCRS and TargetCRS
elements, used by some WCTS operation requests. </documentation>
       </annotation>
       <sequence>
          <element name="SourceCRS" type="anyURI">
             <annotation>
                <documentation>The coordinate reference system (CRS)
used by coordinates input to a Transform operation. This element shall
uniquely identify the desired CRS, but the definition of that CRS need
not be known to the WCTS server. This element is normally a reference
to that CRS, but can contain the definition of that CRS.
</documentation>
             </annotation>
          </element>
          <element name="TargetCRS" type="anyURI">
             <annotation>
                <documentation>The coordinate reference system (CRS)
used by coordinates output from a Transform operation. This element
shall uniquely identify the desired CRS, but the definition of that CRS
need not be known to the WCTS server. This element is normally a
reference to that CRS, but can contain the definition of that CRS.
</documentation>
             </annotation>
          </element>
       </sequence>
    </group>
    <!-- =========================================================== -->
    <element name="Transformation"
type="gml:CoordinateOperationRefType">
       <annotation>
          <documentation>A coordinate operation that can transform
coordinates from the sourceCRS to the targetCRS identified in an
operation request. This element is often a reference to a well-known
coordinate operation, but can contain the definition of a coordinate
operation that references a well-known operation method. Alternately,
this element can contain the definition of a ConcatenatedOperation that
combines two or more well-known coordinate operations or coordinate
operations that reference a well-known operation method. The well-known
coordinate operation or operation method can be defined in the CRS
application profile referenced by the WCTS Implementation
Specification. Alternately or in addition, the well-known coordinate
operation or operation method can be defined in the Capabilities XML
document available from this WCTS server. (TBR) </documentation>
       </annotation>
    </element>
    <!-- =========================================================== -->
    <simpleType name="ServiceType">
       <annotation>
          <documentation>Service type identifier, where the string
value is the OGC Web Service type abbreviation. </documentation>
       </annotation>
       <restriction base="string"/>
    </simpleType>
    <!-- ========================================================= -->
    <simpleType name="VersionType">
       <annotation>
```

```
          <documentation>The version of the Implementation
Specification (document) to which the requested operation conforms. The
value shall be N.N.N, where each N is a non-negative integer
up to 99. </documentation>
       </annotation>
       <restriction base="string"/>
    </simpleType>
    <!-- ========================================================== -->
    <simpleType name="GeometryTypeType">
       <annotation>
          <documentation>Type of GML 3 geometric primitive possibly
handled by a WCTS server. The possible values are all names of GML 3
complexTypes (TBR). </documentation>
       </annotation>
       <restriction base="string">
          <enumeration value="Envelope"/>
          <enumeration value="Point"/>
          <enumeration value="LineString"/>
          <enumeration value="Polygon"/>
          <enumeration value="LinearRing"/>
          <enumeration value="Curve"/>
          <enumeration value="LineStringSegment"/>
          <enumeration value="ArcString"/>
          <enumeration value="Arc"/>
          <enumeration value="Circle"/>
          <enumeration value="ArcStringByBulge"/>
          <enumeration value="ArcByBulge"/>
          <enumeration value="ArcByCenterPoint"/>
          <enumeration value="CircleByCenterPoint"/>
          <enumeration value="CubicSpline"/>
          <enumeration value="BSpline"/>
          <enumeration value="Bezier"/>
          <enumeration value="OrientableCurve"/>
          <enumeration value="Surface"/>
          <enumeration value="PolygonPatch"/>
          <enumeration value="Triangle"/>
          <enumeration value="Rectangle"/>
          <enumeration value="Ring"/>
          <enumeration value="OrientableSurface"/>
          <enumeration value="Solid"/>
          <enumeration value="CompositeCurve"/>
          <enumeration value="CompositeSurface"/>
          <enumeration value="CompositeSolid"/>
       </restriction>
    </simpleType>
    <!-- ========================================================== -->
    <simpleType name="CoverageTypeType">
       <annotation>
          <documentation>Type of GML 3 coverage possibly handled by a
WCTS server. For coverages which use specific geometric primitives, a
client should also check if the corresponding geometric primitive types
are supported. </documentation>
       </annotation>
       <restriction base="string">
          <enumeration value="MulitPoint">
             <annotation>
                <documentation>TBD. </documentation>
             </annotation>
          </enumeration>
          <enumeration value="MultiSurface">
             <annotation>
                <documentation>TBD. </documentation>
             </annotation>
          </enumeration>
          <enumeration value="RectifiedGrid">
             <annotation>
                <documentation>TBD. </documentation>
             </annotation>
          </enumeration>
       </restriction>
    </simpleType>
    <!-- ========================================================= -->
    <simpleType name="InterpolationMethodType">
       <annotation>
          <documentation>Codes that identify interpolation methods. The
meanings of these codes are defined in Annex B of ISO 19123: Geographic
information â€” Schema for coverage geometry and functions.
</documentation>
       </annotation>
       <restriction base="string">
          <enumeration value="nearest neighbor"/>
          <enumeration value="bilinear"/>
          <enumeration value="bicubic"/>
          <enumeration value="lost area"/>
          <enumeration value="barycentric"/>
          <enumeration value="none">
             <annotation>
                <documentation>No interpolation. </documentation>
             </annotation>
          </enumeration>
       </restriction>
    </simpleType>
</schema>
```

8.3 TBD operation response

8.3.1 Normal response parameters

The normal response to a valid TBD operation request shall be TBD. More precisely, a response from the TBD operation shall include the parts listed in Table 9. This table also specifies the UML model data type plus the multiplicity and use of each listed part.

Table 9 — Parts of TBD operation response

Name	Definition	Data type and use	Multiplicity and use
a			

NOTE The UML class diagram contained in Subclause C.TBD provides a graphical view of the contents of the TBD operation response listed in Tables 9- TBD.

8.3.2 Normal response XML encoding

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

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns:wcts="http://www.opengeospatial.net/wcts"
xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.opengeospatial.net/wcts"
elementFormDefault="qualified" xml:lang="en">
   <annotation>
      <appinfo>isTransformableResponse.xsd 2004/10/27</appinfo>
      <documentation>This XML Schema encodes the WCTS IsTransformable
operation message. </documentation>
   </annotation>
   <!-- ==============================================================
      elements and types
   ============================================================== -->
   <element name="IsTransformableResponse">
      <annotation>
         <documentation>Response to a valid IsTransformable operation
request sent to a WCTS. </documentation>
      </annotation>
      <complexType>
         <sequence/>
         <attribute name="transformable" type="boolean"
use="required">
            <annotation>
               <documentation>Indicates whether this WCTS server can
perform a transformation from the sourceCRS to the targetCRS identified
in the operation request. The value shall be "true" or "false".
</documentation>
            </annotation>
         </attribute>
         <attribute name="problem" type="wcts:ProblemType"
use="optional">
            <annotation>
               <documentation>Type of transformation problem detected
by WCTS server. This attribute shall be included whenever the
"transformable" attribute is false. </documentation>
```

```
            </annotation>
         </attribute>
      </complexType>
   </element>
   <!-- ============================================================ -->
   <simpleType name="ProblemType">
      <annotation>
         <documentation>Type of transformation problem by WCTS server.
</documentation>
      </annotation>
      <restriction base="string">
         <enumeration value="source CRS">
            <annotation>
               <documentation>WCTS server cannot transform from
identified source CRS. </documentation>
            </annotation>
         </enumeration>
         <enumeration value="target CRS">
            <annotation>
               <documentation>WCTS server cannot transform to
identified target CRS from identified source CRS. </documentation>
            </annotation>
         </enumeration>
         <enumeration value="geometry type">
            <annotation>
               <documentation>WCTS server cannot transform one or more
identified geometry types. </documentation>
            </annotation>
         </enumeration>
         <enumeration value="coverage type">
            <annotation>
               <documentation>WCTS server cannot transform one or more
identified coverage types. </documentation>
            </annotation>
         </enumeration>
         <enumeration value="interpolation method">
            <annotation>
               <documentation>WCTS server cannot perform one or more
identified interpolation methods. </documentation>
            </annotation>
         </enumeration>
         <enumeration value="other">
            <annotation>
               <documentation>WCTS server cannot perform identified
transformation due to some other problem, including incompatibility
between identified parameters. </documentation>
            </annotation>
         </enumeration>
      </restriction>
   </simpleType>
</schema>
```

8.3.3 TBD exceptions

When a TBD server encounters an error while performing a TBD 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 TBD 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
TBD	TBD	TBD

8.4 Examples

A TBD operation request for TBD can look like this encoded in XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<IsTransformable xmlns="http://www.opengeospatial.net/wcts"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengeospatial.net/wcts
isTransformable.xsd" service="WCTS" version="0.0.0">
   <!-- Primary editor: Arliss Whiteside. Last updated 2003/12/02. -->
   <!-- This template can be used for all CRSs defined by the EPSG. -->
   <sourceCRS xlink:href="urn:ogc:srs:EPSG::4326"/>
   <targetCRS xlink:href="urn:ogc:srs:EPSG::23032"/>
   <geometryType>LineStringType</geometryType>
</IsTransformable>
```

An example response to a TBD operation request is:

```
<?xml version="1.0" encoding="UTF-8"?>
<IsTransformableResponse xmlns="http://www.opengeospatial.net/wcts"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengeospatial.net/wcts
isTransformableResponse.xsd" transformable="true"/>
<!-- Primary editor: Arliss Whiteside. Last updated 2003/12/02. -->
```

9 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.

9.1 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="EnvironmentVariable">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:any/>
        </xsd:sequence>
    </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 makes it so that external SLD documents can 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 over top of previous styled layers.

9.2 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 9.3 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>
    <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 style**s (**UserStyle **in th**e Schema) are discussed in Clause TBD.

9.3 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 6.6, 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:_Feature" maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
```

The **UserLayer** also has a **Name** element which can be used to give a name to a user-defined layer so that it could potentially be stored into a capable WMS, which is discussed in Clause TBD. A WCS is used similarly to a WFS in SLD, although WCS usage is considered "experimental". 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 6.6.

9.3.1 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 **FeatureTypeConstraint**s may be used to define the features of a layer, though all **FeatureTypeConstraint**s in a **UserLayer** must come 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-namespace 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.

9.3.2 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 TBD.

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: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-namespace 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.

9.4 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:FeatureStyle"/>
                <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 6.4). 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 **FeatureStyles** 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 **FeatureStyle** 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 **FeatureStyle**s 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 **FeatureStyle**s 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:FeatureStyle>
                [...]
            </se:FeatureStyle>
        </UserStyle>
    </NamedLayer>
</StyledLayerDescriptor>
```

Annex A
(normative)

Abstract test suite

A.1 General

A paragraph.

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

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.opengeospatial.net/(TBD) where a lower level directory is used for this Version 1.1.1. 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

This annex can be included if useful to provide more XML document examples.

D.1 Introduction

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

D.2 TBD