Open Geospatial Consortium Inc.

Date: 2007-08-14

Reference number of this document: **OGC 07-057r2**

Version: 0.3.0

Category**: OpenGIS® Discussion Paper**

Editor: Keith Pomakis (CubeWerx Inc.)

OpenGIS® Tiled WMS Discussion Paper

Warning

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

Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Contents

i Preface

This document explains how the Web Map Service specification can be extended to allow fast response to a predefined set of tiled maps. It should be read in conjunction with the latest version WMS specification.

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

ii Submitting organizations

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

Autonomous University of Barcelona

CREAF

CubeWerx Inc.

iii Document Contributor Contact Points

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

Name	Organization	E-mail Address
Keith Pomakis	CubeWerx Inc.	pomakis at cubewerx dot com
Joan Masó	UABCREAF	joan.maso at uab dot cat
Núria Julià	UABCREAF	n.Julia at creaf dot uab dot cat

iv Revision history

Date	Release	Editors	Description
2007-05-08	07-057	Joan Masó and Núria Julià	First draft as a discussion paper in the WMS.RWG
2007-06-15	07-057r1	Joan Masó	Corrections proposed mainly by Dimitri Monie and Benjamin Chartier
2007-08-14	07-057r2	Keith Pomakis	Merged OGC 07-057r1 and OGC 07-085r1 into a unified document

v Changes to the OpenGIS® Abstract Specification

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

vi Changes to OpenGIS® Implementation Specifications

Improvements in this document in the context of WMS RWG debate will eventually conduct to a WMS complementary specification candidate like Styled Layer Descriptor Profile of the Web Map Service Implementation Specification (OGC 05-078). Also, a change request to the WMS specification will be submitted in order to allow GetCapabilities to expose which layers will be available as tiled layers and to make GetMap an optional request.

Foreword

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 Web Map Server (WMS) specification was developed to allow cartographic maps to be served over the internet in an interoperable manner. One of the primary goals of the WMS specification (other than interoperability) was flexibility. And in this respect it has been very successful. A WMS client is able to request the overlay of an arbitrary number of map layers in an arbitrary bounding window with an arbitrary background color at an arbitrary scale in a number of coordinate systems with a number of styles (and in some cases, with arbitrary user-defined styles). However, with this flexibility comes a price. Since a WMS server is required to generate each requested map image on the fly, it does not scale well. It is theoretically possible for a WMS to cache the map images that it generates, but in practice, due to the unbounded nature of map requests, this doesn't aid performance much if at all because cache hits will be rare. Many implementations have demonstrated that it is possible to serve most cartographic layer requests within a few seconds, but this is generally in a single-request-at-a-time environment. By the very nature of what a WMS server is required to do, it is unreasonable to expect a WMS server to be able to handle the load of dozens or hundreds of simultaneous requests. Therefore, WMS servers will never be able to achieve the popularity or ubiquity of what the industry is beginning to demand.

In an attempt to tackle this issue, this paper defines an alternative type of WMS operation that uses a tiling model. By limiting the rendering parameters in the request to a discreet set of values, it becomes possible for a WMS server to serve pre-rendered images (called tiles) that require no processing whatsoever. The WMS server can simply return the appropriate pre-generated PNG file. This mechanism is highly scalable (as Google Maps illustrates).

This paper describes two basic operations as proposed extensions to the WMS specification. The **DescribeTiles** operation provides the client with a description (i.e, the available layers, styles, image formats, coordinate systems, scales, sizes and locations) of the image tiles that are available to be requested from the server. The **GetTile** operation provides the client with the ability to request a particular tile. The difference between the standard GetMap operation and the proposed GetTile operation is fundamentally one of flexibility versus scalability. The GetMap operation will continue to exist (as an optionally-supported operation) for those clients that require flexibility over scalability.

OpenGIS® Tiled WMS Discussion Paper

1 Scope

This OpenGIS® document specifies the operations needed to extend the WMS specification to include the ability to request tiled map images (which will usually be pre-generated). It is thus defined as a profile of the WMS, as illustrated in Figure 1.

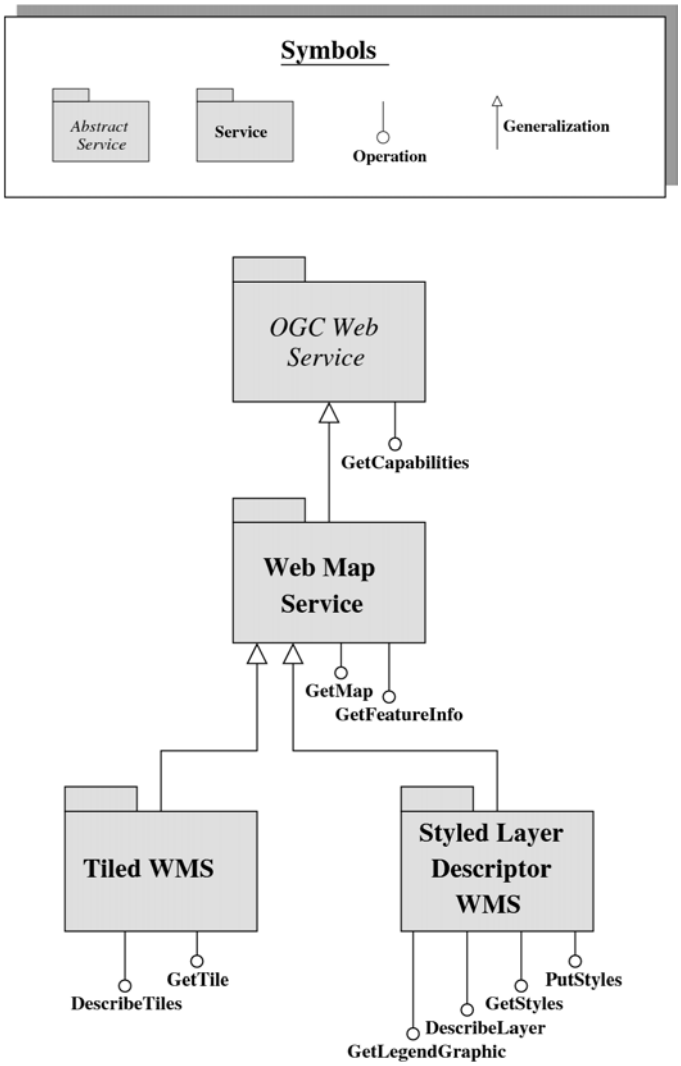

Figure 1: Relation of Tiled WMS to WMS Specification

2 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.

OGC 06-042 (March 2006), *OpenGIS Web Map Server Implementation Specification, version 1.3.0*, Jeff de la Beaujardiere, ed., <http://portal.opengeospatial.org/files/?artifact_id=14416>

OGC 06-121r3 (February 2007), *OGC Web Services Common Specification, version 1.1.0 with Corrigendum 1*, Arliss Whiteside, ed., <http://portal.opengeospatial.org/files/?artifact_id=20040>

XML 1.0 (October 2000), *Extensible Markup Language (XML) 1.0 (2nd edition)*, World Wide Web Consortium Recommendation, Bray, T., Paoli, J., Sperberg-McQueen, C.M., and Maler, E., eds., <http://www.w3.org/TR/REC-xml/>

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

3.1
client
software component that can invoke an **operation** from a **server**

3.2
coordinate reference system
coordinate system that is related to the real world by a datum

3.3
coordinate system
set of mathematical rules for specifying how coordinates are to be assigned to points

3.4
geographic information
information concerning phenomena implicitly or explicitly associated with a location relative to the Earth

3.5
interface
named set of **operations** that characterize the behaviour of an entity

3.6
layer
basic unit of **geographic information** that may be requested as a **map** from a **server**

3.7
map
portrayal of geographic information as a digital image file suitable for display on a computer screen

3.8
operation
specification of a transformation or query that an object may be called to execute

3.9
portrayal
presentation of information to humans

3.10
request
invocation of an **operation** by a **client**

3.11
response
result of an operation returned from a server to a client

3.12
server
a particular instance of a **service**

3.13
service
distinct part of the functionality that is provided by an entity through **interfaces**

3.14
service metadata
metadata describing the **operations** and **geographic information** available at a **server**

4 Conventions

4.1 Requirement levels

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in RFC 2119.

4.2 Abbreviated terms

The following symbols and abbreviations are used in this document:

HTTP	Hypertext Transfer Protocol
MIME	Multipurpose Internet Mail Extensions
PNG	Portable Network Graphics

URL	Uniform Resource Locator
WMS	Web Map Service
XML	Extensible Markup Language

5 Requirements

A tiled WMS service must be scalable. Therefore, servers must be able to quickly return pre-rendered tiles without any image manipulation being necessary.

Support for arbitrary tile sizes, scales and alignments should be possible. This would allow existing tilesets to be served. It would also allow for future adaptation as to which tile sizes, scales and alignments are considered optimal on a project-by-project basis. However, for maximum interoperability, a standard set of tile sizes and scales should be recommended by the tiled WMS specification.

6 Modifications to WMS specification

This discussion paper assumes that the tiled WMS specification will be defined in terms of the next (yet to be defined) WMS specification, and that the version number of that specification will be 1.4.0. The following changes to the WMS specification will be necessary in order to define the tiled WMS profile.

6.1 Definition of LayerType schema type

It is necessary for the WMS schemas to define a LayerType schema type so that the tiled WMS profile can redefine it in order to add an "isTiled" attribute. This can be achieved by changing the following WMS 1.3.0 WMS_Capabilities.xsd schema lines:

```
<element ref="wms:Layer" minOccurs="0"/>

[…]

<element name="Layer">
      <complexType>
            […]
      </complexType>
</element>
```

to:

```
<element name="Layer" type="wms:LayerType minOccurs="0"/>

[…]

<complexType name="LayerType">
      […]
</complexType>
```

6.2 Making the GetMap operation optional

It should be possible to implement a tiled WMS server that only supports tiles without requiring the implementation of all of the tricky dynamic-image-generation logic that is required in order to support the GetMap operation. Unfortunately, the WMS 1.3.0 specification defines GetMap as a required operation. Therefore, it is recommended that the next version of the WMS specification make GetMap an optional operation. This can be achieved by changing the WMS 1.3.0 WMS_Capabilities.xsd schema line:

```
<element ref="wms:GetMap"/>
```

to:

```
<element ref="wms:GetMap" minOccurs="0"/>
```

However, it is likely that WMS 1.4.0 will be based on OWS Common 1.1.0. The OWS Common 1.1.0 schemas provide a more generic way of specifying the supported operations, so as long as the next version of the WMS specification doesn't state that the GetMap operation is required, then the GetMap operation will automatically be optional.

7 WMS GetCapabilities extensions

The WMS GetCapabilities document should be extended in the following ways:

- A boolean "tiled" attribute should be added to the <Layer> element indicating that the layer is tiled.
- The DescribeTiles and GetTile requests should be mentioned in the <Request> element.

This amounts to defining the following schema fragments:

```
<redefine schemaLocation="../../wms/1.4.0/WMS_Capabilities.xsd">
      <complexType name="LayerType">
            <complexContent>
                  <extension base="wms:LayerType">
                        <attribute name="tiled" type="boolean"
                              default="0"/>
                  </extension>
            </complexContent>
      </complexType>
</redefine>

<xsd:element name="DescribeTiles" type="wms:OperationType"
      substitutionGroup="wms:_ExtendedOperation"/>
<xsd:element name="GetTile" type="wms:OperationType"
      substitutionGroup="wms:_ExtendedOperation"/>
```

8 DescribeTiles operation

The DescribeTiles operation allows clients to retrieve service metadata about the tiles that are available for a specified subset of (or all of) the tiled layers.

8.1 DescribeTiles request

The DescribeTiles request accepts (apart from the standard SERVICE, VERSION and REQUEST parameters) an optional LAYERS parameter whose value is a comma-separated list of the names of the layers whose tile descriptions are requested. If this parameter is omitted, then it shall be considered a request for the tile descriptions of every tiled layer.

It also accepts an optional UPDATESEQUENCE parameter, whose value and meaning is the same as that of the GetCapabilities request.

E.g., the following retrieves the tile descriptions for the coastlines and roads layers:

```
baseUrl?SERVICE=WMS
       &VERSION=1.4.0
       &REQUEST=DescribeTiles
       &LAYERS=coastlines,roads
```

while the following retrieves the tile descriptions for all tiled layers on the server:

```
baseUrl?SERVICE=WMS
       &VERSION=1.4.0
       &REQUEST=DescribeTiles
```

It is expected that a tiling client will first make a standard WMS GetCapabilities request, and then make a DescribeTiles request to determine the tiling capabilities of the server.

8.2 DescribeTiles response

8.2.1 Example DescribeTiles response

The response to a DescribeTiles request looks like this:

```
<WMS_DescribeTilesResponse version="1.4.0">
      <TiledLayer name="coastlines">

      <TiledStyles>
            <Value>darkBlue</Value>
            <Value>thickAndRed</Value>
      </TiledStyles>

      <TiledFormats>
            <Value>image/png</Value>
            <Value>image/gif</Value>
      </TiledFormats>

      <TiledDimension name="TIME">
            <Value>2007-05</Value>
```

```
      <Value>2007-06</Value>
      <Value>2007-07</Value>
</TiledDimension>

<TiledCrs name="CRS:84">
      <!-- optional bounding box of data -->
      <ows:BoundingBox>...</ows:BoundingBox>
      <TileMatrixSet>
            <TileWidth>256</TileWidth>
            <TileHeight>256</TileHeight>
            <TileMatrix scale="1e6">
                  <!-- top left point of tile -->
                  <!-- matrix bounding box    -->
                  <gml:Point>...</gml:Point>
                  <!-- width and height of  -->
                  <!-- matrix in tile units -->
                  <MatrixWidth>60000</MatrixWidth>
                  <MatrixHeight>50000</MatrixHeight>
            </TileMatrix>
            <TileMatrix scale="2.5e6">
                  <gml:Point>...</gml:Point>
                  <MatrixWidth>9000</MatrixWidth>
                  <MatrixHeight>7000</MatrixHeight>
            </TileMatrix>
            <TileMatrix scale="5e6">
                  <gml:Point>...</gml:Point>
                  <MatrixWidth>3000</MatrixWidth>
                  <MatrixHeight>2000</MatrixHeight>
            </TileMatrix>
            <TileMatrix scale="10e6">
                  <gml:Point>...</gml:Point>
                  <MatrixWidth>900</MatrixWidth>
                  <MatrixHeight>700</MatrixHeight>
            </TileMatrix>
            <TileMatrix scale="25e6">
                  <gml:Point>...</gml:Point>
                  <MatrixWidth>300</MatrixWidth>
                  <MatrixHeight>200</MatrixHeight>
            </TileMatrix>
            <TileMatrix scale="50e6">
                  <gml:Point>...</gml:Point>
                  <MatrixWidth>150</MatrixWidth>
                  <MatrixHeight>120</MatrixHeight>
            </TileMatrix>
            <TileMatrix scale="100e6">
                  <gml:Point>...</gml:Point>
                  <MatrixWidth>40</MatrixWidth>
                  <MatrixHeight>30</MatrixHeight>
            </TileMatrix>
            <TileMatrix scale="250e6">
                  <gml:Point>...</gml:Point>
                  <MatrixWidth>3</MatrixWidth>
                  <MatrixHeight>2</MatrixHeight>
            </TileMatrix>
      </TileMatrixSet>
</TiledCrs>

</TiledLayer>
```

```
</WMS_DescribeTilesResponse>
```

The schema for this response is listed in Annex A.

8.2.2 Description of DescribeTiles response

Each TiledLayer consists of:

- a list of styles that tiles are available for.
- a list of image formats that tiles are available for.
- for each extra dimension, a list of sample values that tiles are available for.
- a list of coordinate systems that tiles are available for.
- for each coordinate system, a "matrix set" consisting of a list of tile matrices of various scales.

Each "tile matrix" of a matrix set describes a particular tilespace. It does this by specifying:

- the width and height of each tile in pixels. (This is specified in parent <TileMatrixSet> element, and thus is constant for all tile matrices within a matrix set.)
- the scale of the tiles as a scale denominator. (The WMS specification defines what a scale denominator is and how to interpret it. It defines the "standardized rendering pixel size" to be 0.28mm.)
- the top left (minx, maxy) point of the bounding box of the tile matrix (i.e., the real-world coordinate of the top left corner of the top left pixel of the top left tile).
- the width and height of the tile matrix in tile units (i.e., how many tiles there are).

Therefore, the number of tile-matrix sets that a tiling WMS server serves for a particular layer is a product of:

$$nTiledStyles \times nTiledFormats \times (nTiledDimension1...) \times nTiledCrss$$

and the number of distinct tiles within a particular coordinate system of of a tile-matrix set is a product of:

$$matrixWidth \times matrixHeight$$

The exact bounding box of each tile matrix is given (albeit indirectly). This is important because scale-sensitive filtering rules may lead to different tile-matrix bounding boxes per scale. Also, the tile-matrix bounding boxes at each scale will usually vary slightly due to pixel alignment, and it is important for the client and server to take this variation into account.

The way in which the exact bounding box for each tile matrix is given guarantees that the bounding box covers an integral number of tiles. Given the top left point of the tile matrix in real-world coordinates (tileMatrixMinX, tileMatrixMaxY), the width and height of the tile matrix in tile units (matrixWidth, matrixHeight), the width and height of a tile in pixels (tileWidth, tileHeight), the coordinate system (scale) and the scale (scaleDenominator), the bottom left corner of the bounding box of a tile matrix can be calculated as follows:

```
pixelSpan = scaleDenominator * 0.28e-3 /
     metersPerUnit(crs);
tileSpanX = tileWidth * pixelSpan;
tileSpanY = tileHeight * pixelSpan;
tileMatrixMaxX = tileMatrixMinX + tileSpanX * matrixWidth;
tileMatrixMinY = tileMatrixMaxY - tileSpanY * matrixHeight;
```

Here are computations to convert from a viewing bounding box in CRS space to a range of tileset indexes, with error control. Some of the values computed above are used:

```
epsilon = 1e-6;

tileMinCol = floor((viewMinX - tileMatrixMinX) /
                   tileSpanX + epsilon);
tileMaxCol = floor((viewMaxX - tileMatrixMinX) /
                   tileSpanX - epsilon);
tileMinRow = floor((tileMatrixMaxY - viewMaxY) /
                   tileSpanY + epsilon);
tileMaxRow = floor((tileMatrixMaxY - viewMinY) /
                   tileSpanY - epsilon);

// to avoid requesting empty tiles...
if (tileMinCol < 0) tileMinCol = 0;
if (tileMaxCol >= matrixWidth) tileMaxCol = matrixWidth-1;
if (tileMinRow < 0) tileMinRow = 0;
if (tileMaxRow >= matrixHeight) tileMaxRow = matrixHeight-
     1;
```

To fetch the tiles, a client would scan through tileMinCol to tileMaxCol and tileMinRow to tileMaxRow, all inclusive. The computation to convert from a tile column/row to CRS X and Y values of the upper-left corner of the tile is:

```
crsX = tileCol * tileSpanX + tileMatrixMinX;
crsY = tileMatrixMaxY - tileRow * tileSpanY;
```

8.2.3 Features of DescribeTiles response

The DescribeTiles response defined above has the following features:

- It is simple and relatively straight-forward.
- Each tile-matrix set (specified by style, format, sample dimensions and CRS) can define its own tile size.

- It uses scale denominators rather than resolution, because scales are unit independent (consistent between coordinate systems) and are standard in the industry.
- There are no other named entities (such as tilepad in 07-057r1) for the server and client to keep track of.
- The exact bounding box for each tile matrix (for each coordinate system for each scale) can be easily and unambiguously calculated.
- The representation of the bounding box for each tile matrix is one that guarantees that the bounding box covers an integral number of tiles.
- The tile width and height of each tile matrix is explicitly given, so the range of relevant tile indexes doesn't have to be calculated.
- Arbitrary tile alignments are allowed. This aids in the support of existing tilesets. If a client requires all of its selected layers to have tiles which are aligned to the tiles of all of the other layers, a simple calculation can be used to verify whether or not two given tile matrices are aligned.
- Non-square pixels are not supported. Allowing support for non-square pixels in the standard WMS has been described by some as being "disastrous" because buggy implementations force the WMS world to avoid dealing with non-square pixels. Furthermore, unlike for a standard WMS, it would be the server of a tiling WMS that would dictate the pixel ratio(s), considerably reducing interoperability.
- Support for extra dimensions is included.

9 GetTile operation

Once a client has processed the Capabilities document and the WMS_DescribeTilesResponse document of a tiling WMS server, it can begin requesting tiles with GetTile requests.

9.1 GetTile request

A GetTile request has the following required parameters (apart from the standard SERVICE, VERSION and REQUEST parameters):

LAYER
STYLE
CRS
FORMAT
any mentioned dimensions
SCALE
TILEROW
TILECOL

and the following optional parameter:

EXCEPTIONS

The values for LAYER, STYLE, CRS, FORMAT, any mentioned dimensions and SCALE must exactly match the options that are specified in the WMS_DescribeTilesResponse document. TILEROW and TILECOL are integer indices indicating a particular tile in the tile matrix. The top left tile is given the index of (0,0). TILEROW increases downwards and TILECOL increases to the right. The EXCEPTIONS parameter has the same meaning as the GetMap parameter of the same name, with the default value being INIMAGE.

Here is an example GetTile request (based on the example DescribeTiles response given in Section 0):

```
baseUrl?SERVICE=WMS
        &VERSION=1.4.0
        &REQUEST=GetTile
        &LAYER=coastlines
        &STYLE=darkBlue
        &CRS=CRS:84
        &FORMAT=image/png
        &SCALE=5e6
        &TIME=2007-06
        &TILEROW=42
        &TILECOL=112
```

9.2 GetTile response

The response to a GetTile request is an image of the specified format, with the width and height that has been advertised for that layer/style/format/CRS combination. Full tiles are always returned, even if the tile indices are outside of the range of the tile matrix (in which case a fully-transparent tile should be returned). The background pixels of a tile should be transparent when possible so that the client can overlay the tiles on top of other map data (likely other tiles).

10 Things the specification should recommend

For maximum interoperability, the tiled WMS specification should make a few strong recommendations as to what all tiled WMS servers should support. (They should be recommendations and not requirements so that the way things are done is not set in stone.)

10.1 A standard set of scales

A standard set of scales should be strongly recommended. One possible such set is:

1e3, 2.5e3, 5e3, 10e3, 25e3, 50e3, 100e3, 250e3, 500e3,
1e6, 2.5e6, 5e6, 10e6, 25e6, 50e6, 100e6, 250e6

These seem to be industry-standard scales. (The pattern is 1, 2.5, 5.) Another possible such set is a powers-of-two set such as what Google Maps uses. More than one set shouldn't be recommended, however, as this would severely compromise interoperability.

A tiled WMS server should be free to choose a minimum scale that is suitable for the data (pre-rendering GTOPO30 at a scale of 1e3 would be extremely wasteful, for example). However, it should be recommended that servers provide tiles for scales up to 250M for compatibility between layers and other servers.

10.2 A standard image format

It should be recommended that tiles be made available in the image/png image format (for the same reason that it's recommended for the GetMap request).

10.3 A standard coordinate system

Tiles from different sources can only be overlayed with each other if they are in the same coordinate system (projection). Therefore, for maximum interoperability, it is important that the tiled WMS specification strongly recommend a coordinate system that tiles be made available in. CRS:84 (WGS84 Geographic) is perhaps a good choice, since it is the most widely supported coordinate system in existing WMS servers. EPSG:3395 (WGS84 World Mercator) may also be a good choice, however, since it does not distort shapes at high and low latitudes like CRS:84 does. If EPSG:3395 is chosen, implementations would have to have a latitude cut-off point (of perhaps ±85°) to avoid the infinite stretch at the poles.

Selecting a standard coordinate system will likely be one of the most contentious aspects of defining a tiled WMS specification, since there is no single ideal coordinate system. The optimum choice of coordinate system varies with the type of data, the spatial area being represented, and the scale. However, for maximum interoperability, it is critical that a single standard coordinate system be recommended for tiled data. Of course a tiled WMS server can always serve a layer in alternative coordinate systems as well if it is deemed useful.

10.4 Multiple baseURL aliases

The specification should recommend that the GetCapabilities document provide multiple baseUrl aliases for the GetTile request, so that a client can cycle through them when making multiple GetTile requests. This is a common technique for tricking web browsers into making a large number of simultaneous requests to the same server, effectively bypassing the browser's built-in throttles which exist to protect against such an outgoing barrage of requests. (NOTE: It's not possible to provide multiple baseUrl aliases in a WMS 1.3.0 capabilities document, but will be possible in WMS 1.4.0, which will presumably be based on OWS Common 1.1.0.)

11 Future work

The next step in defining a tiled WMS server is to write a draft specification and submit it to the OGC for a vote.

In the further future, it is likely that WMS tiling will evolve to be its own service separate and distinct from WMS. If and when this happens, the DescribeTiles response can be extended to become the core of the GetCapabilities response of that new service.

Annex A: WMS_DescribeTilesResponse XML schema

This Annex contains the proposed XML schema for WMS_DescribeTilesResponse. This schema may also be found online at <http://schemas.cubewerx.com/proposedSchemas/tiledWms/WMS_DescribeTilesResponse.xsd>.

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="http://www.opengis.net/wms"
        xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:wms="http://www.opengis.net/wms"
        xmlns:ows="http://www.opengis.net/ows/1.1"
        xmlns:gml="http://www.opengis.net/gml"
        elementFormDefault="qualified">

  <import namespace="http://www.opengis.net/ows/1.1"
          schemaLocation="http://schemas.opengis.ne/ows/1.1.0/owsCommon.xsd"/>

  <import namespace="http://www.opengis.net/gml"

       schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/geometryBasic0d
       1d.xsd"/>

  <!-- ********************************************************************** -->
  <!-- **  The top-level element.                                          ** -->
  <!-- ********************************************************************** -->

  <element name="WMS_DescribeTilesResponse">
    <annotation>
      <documentation>
        A WMS_DescribeTilesResponse is returned in response to a
        DescribeTiles request made on a tiled WMS.
      </documentation>
    </annotation>
    <complexType>
      <sequence>
        <element ref="wms:TiledLayer" minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
      <attribute name="version" type="string" fixed="1.4.0"/>
      <attribute name="updateSequence" type="string"/>
    </complexType>
  </element>

  <!-- ********************************************************************** -->
  <!-- **  The TiledLayer element.                                         ** -->
  <!-- ********************************************************************** -->

  <element name="TiledLayer">
    <annotation>
      <documentation>
        Describes the tiles that are available for a particular WMS layer.
      </documentation>
    </annotation>
    <complexType>
      <sequence>
        <element ref="wms:TiledStyles"/>
        <element ref="wms:TiledFormats"/>
        <element ref="wms:TiledDimension" minOccurs="0" maxOccurs="unbounded"/>
        <element ref="wms:TiledCrs" maxOccurs="unbounded"/>
      </sequence>
    </complexType>
  </element>

  <!-- ********************************************************************** -->
```

```
<!-- **  Value-list elements.                                          ** -->
<!-- ********************************************************************** -->

<element name="TiledStyles">
  <annotation>
    <documentation>
      Describes the list of styles that the tiles of a layer are
      available in.
    </documentation>
  </annotation>
  <complexType>
    <sequence>
      <element ref="wms:Value" maxOccurs="unbounded"/>
    </sequence>
  </complexType>
</element>

<element name="TiledFormats">
  <annotation>
    <documentation>
      Describes the list of image formats (as MIME types) that the tiles
      of a layer are available in.
    </documentation>
  </annotation>
  <complexType>
    <sequence>
      <element ref="wms:Value" maxOccurs="unbounded"/>
    </sequence>
  </complexType>
</element>

<element name="TiledDimension">
  <annotation>
    <documentation>
      Describes the list of sample values of a particular dimension that
      the tiles of a layer are available in.
    </documentation>
  </annotation>
  <complexType>
    <sequence>
      <element ref="wms:Value" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="name" type="wms:NonEmptyString" use="required"/>
  </complexType>
</element>

<element name="Value">
  <simpleType>
    <restriction base="wms:NonEmptyString"/>
  </simpleType>
</element>



<!-- ********************************************************************** -->
<!-- **  The TiledCrs element.                                         ** -->
<!-- ********************************************************************** -->

<element name="TiledCrs">
  <annotation>
    <documentation>
      Describes the set of tile matrices that are available for a
      particular coordinate reference system (CRS).
    </documentation>
  </annotation>
  <complexType>
    <sequence>
      <!-- the (optional) bounding box of the tiled data             -->
      <!-- (in the coordinate system specified by the parent element) -->
      <element ref="ows:BoundingBox" minOccurs="0"/>
      <element ref="wms:TileMatrixSet"/>
```

```
      </sequence>
      <attribute name="name" type="wms:NonEmptyString" use="required"/>
    </complexType>
  </element>

  <!-- ********************************************************************** -->
  <!-- **  The TileMatrixSet element.                                      ** -->
  <!-- ********************************************************************** -->

  <element name="TileMatrixSet">
    <annotation>
      <documentation>
        Describes a particular set of tile matrices.
      </documentation>
    </annotation>
    <complexType>
      <sequence>
        <element name="TileWidth" type="positiveInteger"/>
        <element name="TileHeight" type="positiveInteger"/>
        <element ref="wms:TileMatrix" maxOccurs="unbounded"/>
      </sequence>
    </complexType>
  </element>

  <!-- ********************************************************************** -->
  <!-- **  The TileMatrix element.                                         ** -->
  <!-- ********************************************************************** -->

  <element name="TileMatrix">
    <annotation>
      <documentation>
        Describes a particular tile matrix.
      </documentation>
    </annotation>
    <complexType>
      <sequence>
        <!-- top left point of tile matrix bounding box -->
        <element ref="gml:Point"/>
        <!-- width and height of matrix in tile units -->
        <element name="MatrixWidth" type="positiveInteger"/>
        <element name="MatrixHeight" type="positiveInteger"/>
      </sequence>
      <attribute name="scale" type="double" use="required"/>
    </complexType>
  </element>

  <!-- ********************************************************************** -->
  <!-- **  Miscellaneous elements.                                         ** -->
  <!-- ********************************************************************** -->

  <!-- (This should be defined in OWS Common.) -->
  <simpleType name="NonEmptyString">
    <restriction base="string">
      <pattern value=".*[^\s].*"/>
    </restriction>
  </simpleType>

</schema>
```

Annex B: Discussion of alternatives

This Annex attempts to capture some of the alternative approaches that have been discussed.

1 Overloading the GetMap operation

It has been suggested that instead of defining a separate GetTile operation, the GetMap operation should be overloaded to perform this function. With this approach, the client application would construct a standard GetMap request, but with a bounding box, tile alignment, scale, coordinate system, style, image size and image format that matches the descriptions of the available tiles. When the WMS server receives a GetMap request, it would analyze the parameters, and if it matches a tile, the tile is returned; otherwise it would fall back to performing a full render of the requested map.

The reason this approach was not taken is that the client could never know for certain that a request it is making will be seen as a request for a pre-rendered tile. Roundoff errors in calculations, varying precisions, slight alignment discrepancies, etc., could all be contributing factors in the WMS server not recognizing the request as being for a pre-rendered tile. Clients that implement a tiling GUI could therefore inadvertently trigger full-blown WMS GetMap requests, and there would be no way of detecting this (other than the very poor performance that would result). Conversely, it would be possible for a standard WMS client to inadvertently hit a pre-rendered tile if its parameters were close enough. In this situation the client would not get back exactly the map it requested; the exact scale, location and/or alignment may be slightly shifted. In other words, if the WMS server's criteria for determining whether or not a GetMap request is a request for a pre-rendered tile is too stringent, tiling clients may not get back what they expect; and if it is too lax, non-tiling clients may not get back what they expect. Unless the criteria for determining matches are spelled out very clearly in the specification, and implemented correctly by both the client and the server, there would be no way to guarantee that the client and server are communicating in the intended way.

Another reason for making GetTile a separate request is that in the future it is possible (and perhaps even likely) that the GetTile request may be packaged in its own service separate and distinct from the WMS. It would therefore be wise for it to maintain its own identity.

2 Allowing background color and/or transparency to be specified

It has been suggested that the GetTiles request include BGCOLOR and TRANSPARENT parameters, allowing the client to request a specific background color and/or whether or not the background pixels should be rendered as transparent.

These parameters are not included in the GetTiles request because it would make tiled WMS servers unscalable. In order for a tiled WMS server to honor such parameters, it

would have to perform some image manipulation. At the very least it would have to perform a flood-fill on each image. This does not scale. Furthermore, such parameters are unnecessary. All tiling clients must be able to support the overlaying of images with transparent backgrounds. Furthermore, tiled servers will serve tiles with transparent backgrounds whenever possible. Therefore, if a client wishes to display a particular background color, it merely needs to render that color as a solid background underneath the tiled layers.

A FORMAT parameter is not included for similar reasons.

3 Reporting the background colors in the DescribeTiles response

It has been suggested that the WMS_DescribeTilesResponse document report the background color of the tiles of each layer.

This capability is not included because there is no practical use for such information. All tiling clients must be able to support the overlaying of images with transparent backgrounds. Furthermore, tiled servers will serve tiles with transparent backgrounds whenever possible. Therefore, if a client wishes to display a particular background color, it merely needs to render that color as a solid background underneath the tiled layers. The non-transparent background color of a set of tiles is irrelevant. If raster data happens to exist that has a solid non-transparent background color, but is intended to be overlayed, then it is up to the tiling server to perform a chroma-key transparency fill during the rendering of the tiles for this layer. This should not be the client's responsibility.

4 Allowing more than one layer to be requested

It has been suggested that the GetTile request should accept a comma-separated list of layers, with the behavior that if the specified layers all have the same tile size, alignment and scales, the tiled WMS server should return a single tile that consists of the overlayed tiles of each of the specified layers.

This ability is not included in the GetTiles request because it would make tiled WMS servers unscalable. In order for a tiled WMS server to honor such a request, it would have to merge the appropriate tiles together on-the-fly. Furthermore, the overlaying of tiles should be a client-level operation. Allowing the client to delegate this operation to the server under certain conditions would unnecessarily complicate the protocol.

5 Specifying pixel sizes rather than scales

It has been suggested that the WMS_DescribeTilesResponse document (and therefore the corresponding parameter of the GetTile request) should specify available tile resolutions by indicating pixel sizes rather than scales (in the form of scale denominators). The basis of this suggestion is that conversion between spatial coordinates and tile indices would be simpler.

This discussion paper recommends the use of scales (in the form of scale denominators) since scales are independent of the units of the map projection. This allows a single standard set of recommended scales to be used, across all coordinate systems. If pixel sizes were used instead, a set of recommended scales would have to be defined for each coordinate system.

A secondary reason for using scales is that it is already in common use in the WMS and SLD specifications (as MinScaleDenominator and MaxScaleDenominator elements).

A third reason for using scales is that it is the industry-standard way of specifying the size of a map relative to the size of the area it represents. It is also a common way of specifying the resolution of vector layers. (Although, admittedly, pixel size is often used to specify the resolution of raster layers). Adjusting the scale number by the ratio of the display's actual pixel size and the "standard" pixel size of 0.28mm allows very accurate scales to be reported to and specified by the user.

The conversion between scale and pixel size for a particular coordinate system is trivial, so the complexity of the calculations involved it not an issue either way.