

CHANGE REQUEST

▪ **OWS Common CR 09-011** ▪ rev **1** ▪ Current version: **1.2.0** ▪

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ▪ symbols.*

Proposed change affects: ▪ AS Imp Spec Best Practices Paper Other

Title:	▪ OWS Common change request – Allow alternative URLs for operation HTTP endpoints		
Source:	▪ Keith Pomakis (pomakis@cubewerx.com)		
Work item code:		Date:	▪ 2009-01-22
Category:	▪ C		
<p><i>Use <u>one</u> of the following categories:</i></p> <ul style="list-style-type: none"> F (Critical correction) A (corresponds to a correction in an earlier release) B (Addition of feature), C (Functional modification of feature) D (Editorial modification) <p>Detailed explanations of the above categories can be found in the TC Policies and Procedures.</p>			

Reason for change:	▪ A Capabilities document needs to be able to provide alternative URLs for the endpoints of its operation requests in order to give browser-based client software the ability to make many requests to the server at once (bypassing the browser's built-in throttles). A specific use case for this requirement is the WMTS GetTile request.
Summary of change:	▪ Change the multiplicity of the URL element within DCP/HTTP/Get and DCP/HTTP/Post from "One (mandatory)" to "One or more (mandatory)" and specify that if multiple URLs are provided, they are alternative URLs for what is conceptually the same endpoint.
Consequences if not approved:	▪ Browser-based OWS client software will never be able to deliver the performance that efficient cache-based OWS servers are able to provide. E.g., browser-based WMTS clients will never be able to render maps as fast as Google Maps can.

Clauses affected:	▪	
Other specs Affected:	<input type="checkbox"/> Other core specifications <input type="checkbox"/> Abstract specifications <input type="checkbox"/> Best Practices Document	▪
Supporting Doc.	▪	
Other comments:	▪	
Status	▪	
Disposition	▪	

1 The problem

In order for a browser-based OWS client which makes use of efficient cache-based or file-system-only OWS servers to deliver the performance that the servers are capable of providing, it may need to make several requests to the server at once. For example, for a WMTS client to render two map layers which consist of 5x4 (20) tiles each, it must make 40 GetTile requests to the server. Since WMTS servers tend to serve pre-rendered tiles, they are usually capable of efficiently handling many more than 40 requests at a time. (This is really the whole point of introducing WMTS as an alternative to WMS.)

However, most if not all modern web browsers have a built-in throttle which limits how many simultaneous requests a web page is allowed to make to a specific server. If more than that number of requests are made to the server, the rest are queued up and made one at a time as one of the previously-made requests completes. This number is typically 8 (but is sometimes configurable; in Firefox this configuration parameter is called "network.http.max-connections-per-server"). This throttle exists to protect the world of web servers from being hit with a barrage of requests. This makes sense as a default behavior, since it prevents unintentional abuse of web servers, but is an unnecessary bottleneck in situations where the servers are known to be able to handle many simultaneous requests and the browser-based client software for it relies on making many simultaneous requests in order to deliver the desired responsiveness.

To continue the WMTS example, a browser-based client application that attempts to build a map consisting of the responses of 40 GetTile requests to the same server would start by making 8 GetTile requests, then 1, then 1, then 1, then 1, then 1.... Each request should provide a quick response, but performing them in serial adds considerable overhead to the total time, resulting in several seconds of elapsed time in order to complete a fully-rendered map. Meanwhile, the server and bandwidth is sitting mostly idle with its potential for simultaneous delivery of tiles mostly untapped.

2 The solution

A reasonable way to solve this problem is to allow a server to advertise one or more alternative URLs for each of its operation request endpoints. This allows a client application to iterate over the alternative URLs when preparing the set of requests it wishes to make to the server. This has the effect of tricking the browser into thinking that these requests are being sent to different servers, therefore bypassing the throttle.

E.g., a browser-based WMTS client may actually generate the following GetTile requests:

```
http://alias1.somewhere.com/wmts.cgi?...&REQUEST=GetTile...
http://alias2.somewhere.com/wmts.cgi?...&REQUEST=GetTile...
http://alias3.somewhere.com/wmts.cgi?...&REQUEST=GetTile...
http://alias4.somewhere.com/wmts.cgi?...&REQUEST=GetTile...
http://alias5.somewhere.com/wmts.cgi?...&REQUEST=GetTile...
http://alias1.somewhere.com/wmts.cgi?...&REQUEST=GetTile...
http://alias2.somewhere.com/wmts.cgi?...&REQUEST=GetTile...
```

http://alias3.somewhere.com/wmts.cgi?...&REQUEST=GetTile...
http://alias4.somewhere.com/wmts.cgi?...&REQUEST=GetTile...
http://alias5.somewhere.com/wmts.cgi?...&REQUEST=GetTile...
...

These alternative URLs may be realized as DNS aliases that point to the same IP address, or as URLs to distinct but synchronized servers.

This is actually what Google Maps does in order to generate its client-side maps as quickly as it does. The URLs of the individual tiles that it requests from the Google Maps tile server look like this:

http://mt0.google.com/...
http://mt1.google.com/...
http://mt2.google.com/...
http://mt3.google.com/...
http://mt0.google.com/...
http://mt1.google.com/...
http://mt2.google.com/...
http://mt3.google.com/...
...

A DNS lookup of these names indicates that they all point to the same server (with the canonical name of mt.l.google.com).

3 The current mechanism

The OWS Common specification already includes a rough mechanism to handle this. It allows multiple <Get> and <Post> elements to exist within a single <HTTP> element, and indicates:

Normally, one Get and/or one Post is included in this subsection. More than one Get and/or Post is allowed to support including alternative URLs for uses such as load balancing or backup.

However, this mechanism is insufficient, since each <Get> and <Post> element can represent a conceptually different endpoint, each of which has its own set of constraints. If a server has a number of alternative URLs to report for an endpoint, it's unreasonable for a server to have to repeat the constraint information for that endpoint for each of the alternative URLs, and it's unreasonable to expect a client application to correlate this information.

4 The change request

4.1 Table 16

Change the footnote of Table 16 ("Parts of HTTP data structure") from:

Normally, one Get and/or one Post is included in this subsection. More than one Get and/or Post is allowed to support including alternative URLs for uses such as load balancing or backup.

to:

Normally, one Get and/or one Post is included in this subsection. More than one Get and/or Post is allowed to support connect points with different Constraints.

4.2 Table 17

Change the "Multiplicity and use" of the URL parameter in Table 17 ("Parts of Request Method data structure") from "One (mandatory)" to "One or more (mandatory)" and provide the following footnote:

If multiple URLs are provided, they are alternatives to one another. In other words, they will produce exactly the same response for requests with the same request parameter values. This can be useful for load balancing, backup (if the first URL is detected as being out of service or otherwise inaccessible), or for other uses prescribed by the individual implementation specifications.

4.3 Figures 6 and C.5

Adjust the UML diagrams in Figures 6 and C.5 appropriately to change the multiplicity of the OnlineResource data type from "one" to "one or more".

4.4 Schema

Change the definition of ows:RequestMethodType from:

```
<complexType name="RequestMethodType">
  <complexContent>
    <extension base="ows:OnlineResourceType">
      <sequence>
        <element name="Constraint" type="ows:DomainType"
          minOccurs="0" maxOccurs="unbounded" />
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

to:

```
<complexType name="RequestMethodType">
  <sequence>
    <element name="OnlineResource" type="ows:OnlineResourceType"
      maxOccurs="unbounded">
    <element name="Constraint" type="ows:DomainType"
      minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
</complexType>
```

Note that this replaces an `xlink:href` attribute with an explicit `OnlineResource` element so that a multiplicity of “one or more” is possible.