WMTS 1.0 Styles API Profile Specification

Author: Keith Pomakis (CubeWerx Inc.)
Date: 2019-02-11

NOTE This has been submitted as OGC change request #566.

Introduction

This WMTS 1.0 profile defines a set of RESTful endpoints that provide a client with the ability to GET, PUT and DELETE style definitions, and a KVP GetStyle operation.

Declaration Of Profile

A WMTS server that provides these RESTful endpoints and/or KVP operations must declare the profile URI *[TBD]* in its capabilities document.

Style URL Templates (RESTful)

The RESTful endpoints that provide a client with the ability to GET, PUT, and DELETE style definitions are declared on a per-layer basis through URL templates with a resourceType of "style". These templates must contain exactly one template variable: `{style}`. The following is an example:

Example 1. Style URL Template

```
<ResourceURL format="application/vnd.ogc.sld+xml" resourceType="style"
    template="https://example.com/wmts/1.0.0/layers/MyLayer/{style}.sld"/>
```

A layer should declare a style URL template for each style encoding (format) that it supports, and must not declare any style URL templates if it is unable to provide or accept style definitions for the layer.

If an endpoint is accessed with an HTTP method that it doesn't support, the server must return an HTTP 405 "Method Not Allowed" exception. If the user simply doesn't have the proper authorization to access the endpoint with that method, the server may instead return an HTTP 401 "Unauthorized" exception with the appropriate WWW-Authenticate header.

The following HTTP methods are defined for these endpoints:

GET

Returns the definition of the style with the identifier specified by `{style}` in the encoding declared by the URL template. If the encoding is SLD, an optional sld_version parameter may be used to indicate the desired SLD version. If this parameter is not present, or if the WMTS server does not support the parameter, or if the WMTS server does not support the desired SLD version, then the server should return SLD version 1.1.0.

If the encoding is SLD, the returned SLD must contain exactly one NamedLayer element, which must have the same identifier as the layer that declared the URL template. That NamedLayer element must contain exactly one UserStyle element, which must represent the requested style.

PUT

Sets the definition of the style with the identifier specified by `{style}` in the encoding declared by the URL template. If the layer already has a style with that identifier, then the style's definition is replaced with the newly-supplied one. Otherwise, a style with that identifier is created for the layer.

If the value of an HTTP Content-Type request header is not equivalent to the encoding (format) declared by the URL template, the server must either throw an HTTP 415 (Unsupported Media Type) exception or accept it anyways (if it supports that content type).

If the encoding is SLD, the request body must be an SLD containing exactly one NamedLayer element which must have the same identifier as the layer that declared the URL template. That NamedLayer element must contain exactly one UserStyle element. NamedStyle elements are not supported because they do not provide style definitions. The style must refer to valid feature sets and feature-set properties. If any of these requirements are violated, the server should throw an HTTP 400 (Bad Request) exception. The mechanism for determining the available pool of feature sets and their schemas is currently beyond the scope of the WMTS interface. However, the Limitations And Future Work section below explores two possible approaches.

It may be the case that a server has particular requirements for syntax or structure of its style identifiers. For example, there may be a restriction on the number of characters, or certain characters may be illegal. The server should return an HTTP 400 (Bad Request) exception if an attempt is made to create a style with an invalid identifier. In such a situation, the response body should contain an explanation of why the identifier is invalid.

On success, an HTTP 200 (OK) or HTTP 204 (No Content) status code should be returned.

DELETE

Deletes the style with the identifier specified by `{style}`. On success, an HTTP 200 (OK) or HTTP 204 (No Content) status code should be returned. If no such style existed, the server may instead choose to return a 404 (Not Found) status code.

The WMTS specification requires all layers to have at least one style. This profile does not dictate the behavior when an attempt is made to delete the last style of a layer. Two possibilities are: a) the request is rejected with an HTTP 400 (Bad Request) exception, or b) the layer ceases to be exposed through the WMTS interface.

OPTIONS

Requests which HTTP methods are available for the endpoint. As per standard HTTP practice, the response must include an "Allow" header whose value is a comma-separated list of allows HTTP methods for the endpoint.

Example 2. An "Allow" Response Header

```
Allow: GET,PUT,DELETE,HEAD,OPTIONS
```

A client may use this method to determine which HTTP methods are available for an endpoint. Alternatively, it is acceptable for the client to try the request and accept the possibility of an HTTP 405 "Method Not Allowed" response.

GetStyle Operation (KVP)

In addition to the above RESTful endpoints, this profile defines the following KVP operation.

operation name: **GetStyle**

parameters:

- **layer** *(required)*
- **style** *(required)*
- **format** *(optional; defaults to "application/vnd.ogc.sld+xml")*
- **sld_version** *(optional; relevant only if format is an SLD format; defaults to "1.1.0")*

This operation returns the definition of the specified style of the specified layer in the specified encoding (format).

If the WMTS server supports this operation, it must be declared in the OperationsMetadata section of the capabilities document.

If an SLD encoding is requested, the optional `sld_version` parameter may be used to indicate the desired SLD version. If this parameter is not present, or if the WMTS server does not support the parameter, or if the WMTS server does not support the desired SLD version, then the server should return SLD version 1.1.0.

If an SLD encoding is requested, the returned SLD must contain exactly one NamedLayer element, which must have the same identifier as the layer that declared the URL template. That NamedLayer element must contain exactly one UserStyle element, which must represent the requested style.

Vendor-specific extensions to this operation (such as making the `style` parameter optional to allow the client to request an SLD representing all of the styles of a layer) are allowed.

This profile does not define a PutStyle or DeleteStyle operation, since such operations cannot adequately be implemented as traditional KVP requests.

Security Considerations

The HTTP PUT and DELETE methods should be privileged methods that only certain users are granted the ability to invoke. It may also be the case that the ability to invoke these methods varies from layer to layer, or even from encoding to encoding. That is, a user may have the authorization to define and adjust the styles for some layers but not others. The means to define such access-control rules is server-specific and beyond the scope of the WMTS specification.

Limitations And Future Work

In order to create new styles or to modify existing styles, it is often necessary for the client to know what feature sets are available and what their schemas are. A client can determine some of this information by examining existing styles. However, there does not yet exist a mechanism for determining the full set of available feature sets and their schemas.

A few possible approaches were explored during the OGC VT-Pilot Extension project, but none of them thoroughly enough to provide a basis for a formal proposal. Two promising approaches are:

1. Make use of the GetAssociations operation proposed by the Web Integration Service Engineering Report in OGC Testbed 12. This provides a bridge between a WMTS service and one or more companion WFS and/or WCS services, allowing the client to query the WFS service(s) for the list of available feature sets and their schemas. It also provides other advantages such as alleviating the need to define a transactional WMTS service, since any manipulation of the feature set(s) rendered by a WMTS layer could be performed through the appropriate WFS or WCS service.
2. Define a new WMTS endpoint that returns a list of the feature sets that are available (regardless of whether or not they're actually rendered by any of the existing styles of any of the currently-defined layers), and provide an endpoint for each of those feature sets that returns that feature set's schema. This is a more direct approach, but adds redundancy between the WMTS and the WFS that could be avoided by making better use of the associations between services.