Web Map Tile Service Developer Guide

Cloud Services | August 2013

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 About This Document

This document covers the concepts of Web Map Tile Service (WMTS), Open Geospatial Consortium (OGC) standards for WMTS, capabilities of WMTS and ways to integrate DigitalGlobe Cloud Services (DGCS)-WMTS in GIS-based custom application development.

1.2 Targeted Audience

This document is targeted to help developers of GIS-based custom applications. Developers new to WMTS can read about the DGCS-WMTS framework, capabilities, integration procedures and development best-practices to design methods for creating innovative world-class GIS applications.

1.3 What is WMTS?

The OGC WMTS Implementation Standard provides an interface to serve digital maps using predefined image tiles. The WMTS standard complements the existing Web Map Service (WMS) standard of the OGC. Imagery is always returned in relation to the world-wide grid and the grid is unique per projection. This forces the clients to mosaic the tiles obtained from the server and then clip the set of tiles into a final image.

Definition of a Tile

The tile resource is generally a rectangular image containing cartographic data. Alternatively, this resource might be a non-image representation of the tile such as a description of the tile or a link to the actual image. For example, the tile resource could be a KML document used in a super-overlay, or a tile metadata document. When returning an image tile, a full single tile *will* always be returned. Also, 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 (possibly other tiles).

1.4 References

- http://www.opengeospatial.org/standards
- http://en.wikipedia.org/wiki/GIS#OGC_standards
- http://en.wikipedia.org/wiki/Geography_Markup_Language

2 Geography Markup Language (GML)

2.1 Introduction to GML

The Geography Markup Language (GML) is the XML code used by the OGC to express geographical features. GML serves as a modeling language for geographic systems as well as an open interchange format for geographic transactions on the Internet. Note that the concept of feature in GML is a very general one and includes not only conventional "vector" or discrete objects, but also coverages and sensor data. The ability to integrate all forms of geographic information is key to the utility of GML.

GML was conceived and evolved for a variety of reasons, the most important reasons being:

- To provide a language for expressing geographic entities – to create application specific geographic vocabularies.
- To enable the encoding of geographic information consistent with these vocabularies.
- To support geospatial queries and transactions across the Internet.

GML is feature-centric. Features are entities – things that describe aspects of the real world from the perspective of a particular application community – whether circumscribed by geography or function or both. GML vocabularies are created by communities of interest. These vocabularies are called GML Application Schemas. If you look at such an Application Schema you will find real world objects like Buildings, Roads, Buoys, Navigation Aids, Airline Flight Paths, Vehicles and Railway Switches. Each such object is defined in the schema by listing its properties. For example, a Building might be described by:

```
<abc:Building gml:id='b143'>
      <abc:height>40</abc:height>
      <abc:footprint>
            <gml:polygon></gml:polygon>
      </abc:footprint>
</abc:Building>
```

Note that the Building (feature) has two properties, namely height and footprint. The height property in this case has an integer value (number of stories), while the footprint property has a Polygon (shape) for a value.

GML application schemas can be the basis of standards themselves – such as S57GML, cityGML, geoRSS GML and AIXM, or they can be informal creations for only a very small community. Which is the case is up to the community.

GML application schemas should NOT be confused with GML profiles. A GML profile is a subset of GML, defined usually by the subset tool (part of the GML specification), consisting of selected element, attribute and type declarations and all dependent components from the GML core schemas (the schemas defined by the GML specification). Application schemas can be built on GML profiles. Some GML profiles are also specifications and this includes the GML Simple Features Profile, the Point Profile, the GML Profile for GMLJP2 and the GML Profile for GeoRSS.

GML was developed to support geographic requests and transactions and this usage predates the WCS developed for this purpose. When a user sends a request for geographic data – e.g. “find all water wells within this county” – there must be a way to express “water well”, “county” and the “geometric extend of the county”. In WCS, GML is used for this purpose. When the user wants to send a transaction such as “change the shape of the Holmes River to the following …” they need a way to express the river’s geometry; GML provides this mechanism in the WCS.

2.2 Overview of GML Schema

GML specifies XML encodings of a number of the conceptual classes defined in the ISO 19100 series of International Standards and the OpenGIS Abstract Specification in conformance with these standards and specifications.

In many cases, the mapping from the conceptual classes to XML is straightforward, while in some cases the mapping is more complex.

In addition, GML provides XML encodings for additional concepts not yet modeled in the ISO 19100 series of International Standards or the OpenGIS Abstract Specification. Examples include moving objects, simple observations or value objects. Additional conceptual classes corresponding to these extensions are also specified in Annex D.

The GML schema comprises the components (XML elements, attributes, simple types, complex types, attribute groups, groups, etc.) that are described in this International Standard. The XML encoding conforms to ISO 19118.

2.3 GML Schema Features

A GML feature is a feature encoded using GML. Examples include a road, a river, a person, a vehicle, an administrative area, or an event.

The feature schema provides a framework for the creation of GML features and feature collections.

2.3.1 ABSTRACTFEATURETYPE

The basic feature model is given by the `gml:AbstractFeatureType`, defined in the schema as follows:

```
<complexType name="AbstractFeatureType" abstract="true">
      <complexContent>
            <extension base="gml:AbstractGMLType">
                  <sequence>
                        <element ref="gml:boundedBy" minOccurs="0"/>
                        <element ref="gml:location" minOccurs="0"/>
                  </sequence>
            </extension>
      </complexContent>
</complexType>
```

The content model for `gml:AbstractFeatureType` adds two specific properties suitable for geographic features to the content model defined in `gml:AbstractGMLType`.

The value of the `gml:boundedBy` property describes an envelope that encloses the entire feature instance, and is primarily useful for supporting rapid searching for features that occur in a particular location.

The value of the `gml:location` property describes the extent, position or relative location of the feature. `gml:location` is deprecated as part of the standard content model of `gml:AbstractFeatureType`.

2.3.2 ABSTRACTFEATURE

The element `gml:AbstractFeature` is declared as follows:

```
<element name="AbstractFeature"
           type="gml:AbstractFeatureType"
           abstract="true"
           substitutionGroup="gml:AbstractGML"/>
```

This abstract element serves as the head of a substitution group which may contain any elements whose content model is derived from `gml:AbstractFeatureType`. This may be used as a variable in the construction of content models.

`gml:AbstractFeature` may be thought of as anything that is a GML feature and may be used to define variables or templates in which the value of a GML property is “any feature”. This occurs in particular in a GML feature collection where the feature member properties contain one or multiple copies of `gml:AbstractFeature` respectively.

The Other features which are used are boundedBy, BoundingShapeType, EnvelopeWithTimePeriod, EnvelopeWithTimePeriodType, **locationName, locationReference,FeaturePropertyType, FeatureArrayPropertyType.**

2.4 Overview

Many applications require definitions of terms which are used within instance documents as the values of certain properties or as reference information to tie properties to standard information values in some way. Units of measure and descriptions of measurable phenomena are two particular examples.

It will often be convenient to use definitions provided by external authorities. These may already be packaged for delivery in various ways, both online and offline. In order that they may be referred to from GML documents it is generally necessary that a URI be available for each definition. Where this is the case then it is usually preferable to refer to these directly.

Alternatively, it may be convenient or necessary to capture definitions in XML, either as a separate document or embedded within an instance document containing features. The definitions may be transcriptions from an external source, or may be new definitions for a local purpose. In order to support this case, some simple components are provided in GML in the form of:

- A generic `gml:Definition`, which may serve as the basis for more specialized definitions.
- A generic `gml:Dictionary`, which allows a set of definitions or references to definitions to be collected.

These components may be used directly, but also serve as the basis for more specialized definition elements in GML, in particular: coordinate operations (Clause 12), coordinate reference systems (Clause 12), datums (Clause 12), temporal reference systems (Clause 14), and units of measure (Clause 16).

Note that the GML definition and dictionary components implement a simple nested hierarchy of definitions with identifiers. The latter provide handles which may be used in the description of more complex relationships between terms. However, the GML dictionary components are not intended to provide direct support for complex taxonomies, ontologies or thesauri. Specialized XML tools are available to satisfy the more sophisticated requirements.

➔ The dictionary schema document is identified by the following location-independent name (using URN syntax): urn:x-ogc:specification:gml:schema-xsd:dictionary:3.2.1.

2.4.1 DICTIONARY SCHEMA

Definition, DefinitionType, remarks

The basic `gml:Definition` element specifies a definition, which can be included in or referenced by a dictionary. It is declared as follows:

```
<element name="Definition" type="gml:DefinitionType"
substitutionGroup="gml:AbstractGML"/>
 <complexType name="DefinitionBaseType">
    <complexContent>
        <restriction base="gml:AbstractGMLType">
             <sequence>
                <element ref="gml:metaDataProperty" minOccurs="0"
maxOccurs="unbounded"/>
                <element ref="gml:description" minOccurs="0"/>
                <element ref="gml:descriptionReference" minOccurs="0"/>
                <element ref="gml:identifier"/>
                <element ref="gml:name" minOccurs="0" maxOccurs="unbounded"/>
              </sequence>
               <attribute ref="gml:id" use="required"/>
         </restriction>
    </complexContent>
</complexType>
<complexType name="DefinitionType">
    <complexContent>
          <extension base="gml:DefinitionBaseType">
             <sequence>
               <element ref="gml:remarks" minOccurs="0"/>
             </sequence>
          </extension>
    </complexContent>
</complexType>
<element name="remarks" type="string"/>
```

The content model for a generic definition is a derivation from `gml:AbstractGMLType`.

The gml:description property element shall hold the definition if this can be captured in a simple text string, or the gml:descriptionReference property element may carry a link to a description elsewhere.

The gml:identifier element shall provide one identifier identifying this definition. The identifier shall be unique within the dictionaries using this definition.

The gml:name elements shall provide zero or more terms and synonyms for which this is the definition.

The gml:remarks element shall be used to hold additional textual information that is not conceptually part of the definition but is useful in understanding the definition.

Dictionary, DictionaryType

Sets of definitions may be collected into dictionaries or collections. These are declared in the schema as follows:

```
<element name="Dictionary" type="gml:DictionaryType"
substitutionGroup="gml:Definition"/>
<complexType name="DictionaryType">
      <complexContent>
            <extension base="gml:DefinitionType">
                  <choice minOccurs="0" maxOccurs="unbounded">
                        <element ref="gml:dictionaryEntry"/>
                        <element ref="gml:indirectEntry"/>
                  </choice>
                  <attributeGroup ref="gml:AggregationAttributeGroup"/>
            </extension>
      </complexContent>
</complexType>
```

A gml:Dictionary is a non-abstract collection of definitions.

The gml:Dictionary content model adds a list of gml:dictionaryEntry and *gml:indirectEntry (deprecated)* properties that contain *or reference* gml:Definition objects. A database handle (gml:id attribute) is required, in order that this collection may be referred to. The standard gml:identifier, gml:description, gml:descriptionReference and gml:name properties are available to reference or contain more information about this dictionary. The gml:description and gml:descriptionReference property elements may be used for a description of this dictionary. The derived gml:name element may be used for the name(s) of this dictionary.

dictionaryEntry, DictionaryEntryType

These elements contain or refer to the definitions which are members of a dictionary. The element gml:dictionaryEntry is declared as follows:

```
<element name="dictionaryEntry" type="gml:DictionaryEntryType"/>
<complexType name="DictionaryEntryType">
      <complexContent>
            <extension base="gml:AbstractMemberType">
                  <sequence minOccurs="0">
                        <element ref="gml:Definition"/>
                  </sequence>
                  <attributeGroup ref="gml:AssociationAttributeGroup"/>
            </extension>
      </complexContent>
</complexType>
```

The content model follows the standard GML property pattern, so a gml:dictionaryEntry may either contain or refer to a single gml:Definition. Since gml:Dictionary is substitutable for gml:Definition, the content of an entry may itself be a lower-level dictionary.

Note that if the value is provided by reference, this definition does *not* carry a handle (gml:id) in this context, so does *not* allow external references to this specific definition in this context. When used in this way the referenced definition will usually be in a dictionary in the same XML document.

Using Definitions and Dictionaries

Dictionaries and definitions are GML objects, so may be found in independent GML data instance documents.

In application schemas it might be useful to attach a `gml:Dictionary` or `gml:Definitions` to a feature collection in order to record definitions used in properties of members of the collection.

The following example shows two instances of dictionaries:

```
<gml:Dictionary gml:id="rockTypes">
      <gml:description>
            A simple dictionary of rock types using components from gmlBase
      </gml:description>
      <gml:identifier codeSpace="http://www.abc.org/terms">
             Rock Types
      </gml:identifier>
      <gml:dictionaryEntry>
            <gml:Definition gml:id="granite">
                  <gml:description>
                        A igneous rock normally composed of quartz, two feldspars and
                        optional mica
                  </gml:description>
                  <gml:identifier codeSpace="http://www.abc.org/terms">
                           Granite
                   </gml:identifier>
            </gml:Definition>
      </gml:dictionaryEntry>
      <gml:dictionaryEntry>
            <gml:Definition gml:id="sst">
                  <gml:description>
                           A detrital sedimentary rock normally composed of siliceous
grains
                   </gml:description>
                  <gml:identifier codeSpace="http://www.abc.org/terms">
                            Sandstone
                  </gml:identifier>
            </gml:Definition>
      </gml:dictionaryEntry>
      <gml:dictionaryEntry xlink:href="http://my.big.org/definitions/geology/limestone"/>
</gml:Dictionary>
<gml:Dictionary gml:id="AbridgedGMLdictionary">
<gml:identifier codeSpace="http://www.opengis.net/gml/3.2">
GML Dictionary
</gml:identifier>
<gml:dictionaryEntry>
      <gml:Definition gml:id="term4.1">
            <gml:description>
conceptual schema for data required by one or more applications
</gml:description>
            <gml:identifier codeSpace="http://www.isotc211.org/19101">
application schema
</gml:identifier>
      </gml:Definition>
</gml:dictionaryEntry>
<gml:dictionaryEntry>
      <gml:Definition gml:id="term4.2">
            <gml:description>
application schema written in XML Schema in accordance with the rules specified in ISO
19136
</gml:description>
<gml:identifier codeSpace="http://www.opengis.net/gml/3.2">

Continued…
```

```
…Continued

GML application schema
</gml:identifier>
          </gml:Definition>
</gml:dictionaryEntry>
<gml:dictionaryEntry>
      <gml:Definition gml:id="term4.3">
            <gml:description>
semantic relationship between two or more classifiers that specifies connections
among their instances
GML application schema
</gml:identifier>
          </gml:Definition>
</gml:dictionaryEntry>
<gml:dictionaryEntry>
      <gml:Definition gml:id="term4.3">
            <gml:description>
semantic relationship between two or more classifiers that specifies connections
among their instances
</gml:description>
            <gml:identifer
codeSpace="http://www.uml.org/1.3">association</gml:identifier>
      </gml:Definition>
</gml:dictionaryEntry>
<gml:dictionaryEntry>
      <gml:Definition gml:id="term4.4">
            <gml:description>name-value pair contained in an
element</gml:description>
            <gml:identifer
codeSpace="http://www.w3.org/XML/1998/namespace">attribute</gml:identifier>
      </gml:Definition>
</gml:dictionaryEntry>
<!--.. -->
</gml:Dictionary>
```

3 Web Map Tile Service (WMTS)

3.1 Introduction to WMTS

The DigitalGlobe Web Map Tile Service (WMTS) defines a set of functions that clients may use to return actual features with geometry and attributes that can be used in any type of geospatial analysis.

Any client, making requests that conform to the OGC WMTS specification can interact with the DGCS WMTS server. Web-based client-server architecture is a typical example of the structure of a Web Map Tile Service application, as illustrated in Figure 3.1.

FIGURE 3.1 A TYPICAL STRUCTURE OF A DGCS-WMTS APPLICATION

In the DGCS - WMTS scenario, the client application requests desired information from the Web Map Tile Service server. The WMTS server retrieves from the database the appropriate information including capabilities, Tile and FeatureInfo and responds to the request with relevant information.

The goal of providing a WMTS-enabled service is to offer high performance while being scalable. Therefore, servers must be able to return tiles quickly. A good way to achieve that is to use locally stored, pre-rendered tiles that will not require any image manipulation or geo-processing.

The purpose of a WMTS service is to serve maps divided into individual tiles.

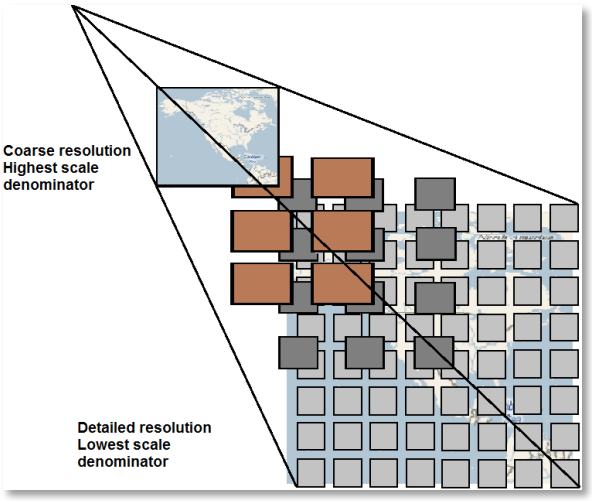

FIGURE 3.2 WMTS SERVICE DIVIDED INTO TILES

3.2 Advantages of WMTS

The DigitalGlobe WMTS provides raster imagery data at multiple resolutions in predefined imagery tiles (PNG or JPEG formats), making it more scalable and high-performing than Web Map Service (WMS). The WMTS is similar to WMS, but it enables better server performance in applications that involve several simultaneous requests. To improve performance, WMTS returns small pre-generated images or reuses identical previous requests that follow a discrete set of tile matrices, rather than creating a new image for each request. This service is optimized for sending rapid image tile delivery information from the cache for display on portals or applications where server response time is of primary concern.

3.3 WMTS Client-Server Architecture

The following architecture depicts a sample integration of WMTS client and server applications. Client Viewer is a series of HTML pages running inside a web browser that can interact with WMTS server via client application through HTTP calls. WMTS client manages the interactions with WMTS interfaces through HTTP requests and dynamically generates HTML that can run in a Web browser.

The WMTS server accepts requests from the WMTS client and the viewer client in the form of HTTP URL strings, and returns results encoded as XML, PNG, GML, and so on.

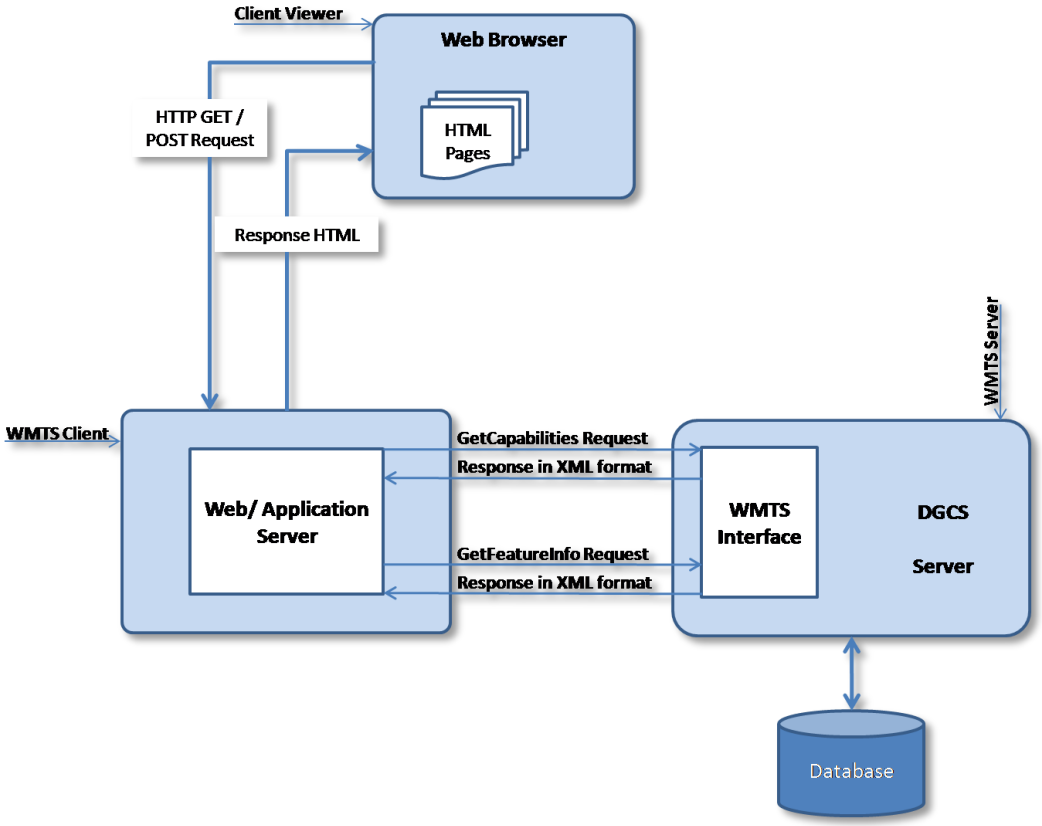

FIGURE 3.3 SAMPLE WMTS CLIENT-SERVER APPLICATION

3.4 WMTS Service Details

The OGC standard for WMTS defines a set of required and optional operations. The DigitalGlobe WMTS supports the following operations:

GetCapabilities

The GetCapabilities request is used to obtain information about the available map tile types and supported operations.

GetTile

The GetTile request is used to obtain an actual imagery tile. Tiles are available for a subset of the layers provided by the online catalogs.

GetFeatureInfo

The GetFeatureInfo request is used to obtain metadata about any tile obtained via the GetTile request.

3.5 Basic Service Elements

This section specifies aspects of Web Map Server behavior that are independent of particular operations or are common to several operations.

3.5.1 HTTP REQUEST

In the client-server computing model, HTTP functions as a request-response protocol. For example, in HTTP, a web browser acts as a client, while an application running on a computer hosting a web site functions as a server. The client submits an HTTP request message to the server. The server, which stores content or provides resources, such as HTML files and images, returns a response message to the client. A response contains completion status information about the request and may contain any content requested by the client in its message body.

An HTTP Uniform Resource Locator (URL) locates the Online Resource of each operation supported by a service instance. The URL may be different for each operation, or the same, at the discretion of the service provider.

HTTP supports two request methods: GET and POST. One or both of these methods may be defined for a particular web service and offered by a service instance. The use of the Online Resource URL differs in each case.

HTTP GET

An Online Resource URL intended for HTTP GET requests, is, in fact, only a URL prefix to which additional parameters must be appended in order to construct a valid Operation request. A URL prefix is defined as an opaque string including the protocol, hostname, optional port number, path, a question mark '?', and, optionally, one or more server-specific parameters ending in an ampersand '&'. The prefix uniquely identifies the particular service instance.

A client can append the necessary request parameters as name/value pairs in the form "name=value&". The resulting URL must be valid according to the HTTP Common Gateway Interface (CGI) standard, which mandates the presence of '?' before the sequence of query parameters and the '&' between each parameter. Table 3.1 summarizes the components of an operation request URL.

The URL prefix must end in either a **'?'** (in the absence of additional server-specific parameters) or a **'&'.** In practice, however, Clients should be prepared to add a necessary trailing **'?'** or '**&'** before appending the operation parameters defined as per DG-WMS specification in order to construct a valid request URL. Please refer to Table 3.2 for reserved characters as per HTTP rules.

TABLE 3.1 A GENERAL GET REQUEST

URL COMPONENT	DESCRIPTION
http://host[:port]/path?{name[=value]&}	URL prefix of service operation. [ ] denotes 0 or 1 occurrence of an optional part; {} denotes 0 or more occurrences. The prefix is entirely at the discretion of the service provider.
name=value&	One or more standard request parameter name/value pairs defined by a web feature service. The actual list of required and optional parameters is mandated for each operation is described in the Table 4.2.

TABLE 3.2 RESERVED CHARACTERS IN HTTP GET QUERY

CHARACTER	RESERVED USAGE
?	Separator indicating start of query string.
&	Separator between parameters in query string.
=	Separator between name and value of parameter
/	Separator between MIME type and subtype in format parameter value.

CHARACTER	RESERVED USAGE
:	Separator between Namespace and Identifier in SRS parameter value.
,	Separator between individual values in list-oriented parameters.

HTTP POST

An Online Resource URL intended for HTTP POST requests is a complete and valid URL to which clients transmit encoded requests in the *body* of the POST document. DGCS-WMTS do not require additional parameters to be appended to the URL in order to construct a valid target for the Operation request. Figure 3.4 shows a sample of an HTTP Post request.

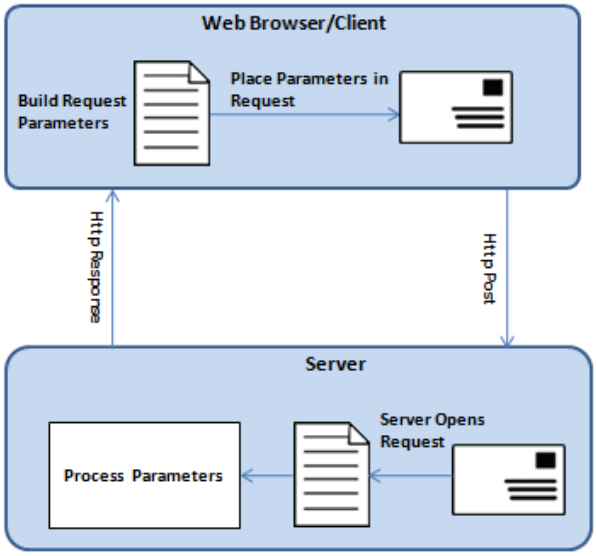

FIGURE 3.4 SAMPLE HTTP POST REQUEST/RESPONSE

Advantages of HTTP Post instead of HTTP Get:

- The Parameter's name and value are visible to the user and to anyone who is looking at the URL in the browser.
- GET requests are passed as the URL string and are therefore limited by the URL length limit specified by the browser.
- HTTP Post method can upload files to the server.

HTTPS

In addition to or instead of offering web map services using the HTTP protocol, DigitalGlobe offers web map service using HTTPS. HTTPS is HTTP over a secure communication channel which allows encrypted information to be transferred between machines over the World Wide Web.

The use of HTTPS does not affect the description of the requests and responses described in this document, but may require additional actions to be taken on both the client and the service in order to initiate secure communication.

3.5.2 HTTP RESPONSE

Upon receiving a valid HTTP request, the service sends a response corresponding to the request exactly as detailed, based on parameters for the specific operations.

Response objects will be accompanied by other HTTP entity headers as appropriate and to the extent possible. In particular, the Expires and Last-Modified headers provide important information for caching; Content-Length may be used by clients to know when data transmission is complete and to efficiently allocate space for results, and Content-

Encoding or Content-Transfer-Encoding may be necessary for proper interpretation of the results. If the request is invalid, the service issues a Service Exception.

Output Formats

The optional “outputFormat” attribute specifies the format of the response to a Cloud Service request. The default value is text/xml; “subtype=gml/3.1.1” indicating that a valid GML3 document, that validates against a valid GML3 application schema, must be generated. For backward compatibility, the values “GML2” or “text/xml; subtype=gml/2.1.2” may be specified indicating that a valid GML2 document that validates against a valid GML2 application schema, must be generated.Table 3.3 summarizes the possible values for the “outputFormat” attribute.

TABLE 3.3 VALUES FOR OUTPUTFORMAT ATTRIBUTE

OUTPUTFORMAT VALUE	DESCRIPTION
GML2	This value is kept for backward compatibility and indicates that an XML instance document must be generated that validates against a GML2 application schema.
text/xml; subtype=gml/2.1.2	Same as GML2.
text/xml; subtype=gml/3.1.1	This value indicates that an XML instance document must be generated that validates against a GML3 application schema. This is the default values of the outputFormat attribute if the attribute is not specified in the GetFeature request.

3.5.3 REQUEST PARAMETERS

As per the specification standards of WMTS, a client application has to form the HTTP(S)-based URL dynamically, based on requirement or operation it has to perform. The following are the list of important parameters which are part of a WMTS URL.

Base URL

For every request to DigitalGlobe WMTS server, client needs to append parameters to the base URL. DigitalGlobe provides the base URL for WMTS, which is used as the common base URL as described below.

Base URL:
https://services.digitalglobe.com/earthservice/wmtsaccess

Username and Password are required only for some accounts. All others require Connect ID.

CONNECTID

ConnectID is a parameter name which needs to be appended along with base URL mentioned above with appropriate value. Value for these parameters is a unique 32 digit alphanumeric value. It is a mandatory parameter which should be part of every request client makes from server. Please contact DigitalGlobe to get your unique ConnectID.

ConnectID format: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
x → alpha numeric number

SERVICE

This parameter defines the type of service the client is requesting. As mentioned above, DigitalGlobe provides different OGC services, including WMS, WFS, WMTS and WCS. Therefore, the client must an provide appropriate value based on the service requested. The value for this parameter always is **“WMTS”** for WMTS clients.

Example: service=WMTS

VERSION

The "**version**" parameter specifies the protocol version number. The version number indicates the specification compliance level as defined by OGC. The format of version number contains three positive integers, separated by decimal points, in the form "**x.y.z"**. The numbers "y" and "z" will never exceed 99. Each Feature Service provided by DigitalGlobe is numbered independently as per respective OGC specification standards. The latest version of DigitalGlobe WMTS implemented for the OGC specification is 1.0.0.

The version number appears in the following two places:

- In response XML of GetCapabilities request describing WMTS service and
- In the parameter list of client requests to the WMTS service.

In response to a GetCapabilities request containing a version number, a WMTS server responds with output that conforms to that version of the specification, or negotiates a mutually agreeable version if the requested version is not implemented on the server. If no version number is specified in the request, the server responds with the highest version it understands and labels the response accordingly. Please refer to OGC specification for WMTS on negotiation rules.

Example: version=1.0.0 (recommended until DigitalGlobe implements newer version per OGC specification)

REQUEST

The REQUEST parameter indicates which service operation is being invoked. The value shall be the name of one of the operations offered by DigitalGlobe WMTS Service. Please refer to Section 3.1 for descriptions of operations supported by DigitalGlobe WMTS.

Example: request=GetCapabilities

FORMAT

The FORMAT parameter specifies the output format of the response to a request operation. Formats are expressed in both Capabilities XML and in operation requests using MIME types. Each Operation has a distinct list of supported formats. Some formats may be offered by several operations, and are then duplicated as needed in each list. If a request contains a Format not offered by WMTS server, the server throws a Service Exception (with code "InvalidFormat").

Example: format=image/jpeg

EXCEPTIONS

The EXCEPTIONS parameter in a request indicates the format in which the Client wants to be notified of Service Exceptions. Individual error messages appear as <ServiceException> elements within the <ServiceExceptionReport> in Service Exception XML. Refer to Section 3.7 on page 21 for more details on service exceptions.

Example: exceptions=application/vnd.ogc.se_xml

3.5.4 REQUEST PARAMETER RULES

While forming request URL, client applications should follow certain rules as described below:

- Parameter names are not case sensitive, but parameter values are case sensitive.
- Parameter names are typically shown in uppercase for typographical clarity, not as a requirement.
- Parameters in a request may be specified in any order.
- When request parameters are duplicated with conflicting values, the response from the server may be undefined.
- Parameters consisting of lists (for example, BBOX, LAYERS and STYLES in WMS GetMap) shall use the comma (",") as the separator between items in the list. Additional white space shall not be used to delimit list items.
- Two successive commas indicate an empty item, as does a leading comma or a trailing comma. An empty list (" ") shall be interpreted either as a list containing no items or as a list containing a single empty item, depending on context.

3.6 Integration Procedure

A WMTS client application is a program that communicates with the DGCS WMTS server using the operations GetCapabilities, GetTile, and, optionally, GetFeatureInfo, as noted earlier. More specifically, in a typical WMTS client-server interaction, the following sequence would be followed:

STEP-1

The client requests **GetCapabilities** from the WMTS server in order to determine what the WMTS server can do and what features the WMTS server can provide.

> Example Request:
>
> https://services.digitalglobe.com/earthservice/wmtsaccess?SERVICE=WMTS&REQUEST=GetCapabilities&VERSION=1.0.0&connectid=<ConnectID>&username=<username>&password=<password>
>
> Username and Password parameters may not be applicable depending on your account type. Replace <ConnectID> with the ConnectID provided by DigitalGlobe. Parameters are not required to be in the same order as shown above.

Understanding URL

The URL shown above contains Base URL and parameters, as explained in Section 3.5 on page 14. The key parameter for this request is "**request=GetCapabilities**", which fetches the capabilities of Web Map Tile Service and responds in the form of XML data.

TABLE 3.4 UNDERSTANDING URL PARAMETERS FOR WMTS GETCAPABILITIES REQUEST

PARAMETER NAME	PARAMETER VALUE	DESCRIPTION
CONNECTID*	**<ConnectID>** provided by DigitalGlobe	The value for this parameter is a unique 32-digit alphanumeric value assigned by DigitalGlobe (explained under CONNECTID in Section 3.5.3). A valid CONNECTID is mandatory for every request.
SERVICE*	WMTS	Refer to SERVICE in Section 3.5.3.
REQUEST*	GetCapabilities	The value for this parameter should always be "**GetCapabilities**" for step-1.
VERSION*	1.0.0	Refer to VERSION in Section 4.5.3.

*Mandatory

Response

In response to a GetCapabilities request, the DGCS WMTS server produces an XML document containing the WMTS server's service metadata, describing all the operations it supports and providing information about available map types. The client application has to parse the XML capabilities document to retrieve the necessary information used to request a feature. The Document Object Model (DOM) is a widely used and efficient XML parser, which can be utilized to parse the XML document and retrieve the information. The DOM represents an XML document as a tree of nodes that can be easily traversed and edited with its standard interfaces.

The response XML to a GetCapabilities request contains the following details:

- WMTS Service details like Name, Title, URL
- Contact Information Person, Organization, Address, Telephone, Fax and Email

WMTS Operations, like GetCapabilties and GetTile, with respective formats and URLs. An example of a GetCapabilities response is shown below:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:Capabilities version="1.0.0" xmlns:ns2="http://www.w3.org/1999/xlink"
xmlns:ns1="http://www.opengis.net/ows/1.1" xmlns:ns3="http://www.opengis.net/wmts/1.0">
      <ns1:ServiceIdentification>
            <ns1:Title>DigitalGlobe Web Map Tile Service</ns1:Title>
            <ns1:ServiceType>OGC WMTS</ns1:ServiceType>
            <ns1:ServiceTypeVersion>1.0.0</ns1:ServiceTypeVersion>
      </ns1:ServiceIdentification>
      <ns1:ServiceProvider>
            <ns1:ProviderName>DigitalGlobe Inc.</ns1:ProviderName>
            <ns1:ProviderSite ns2:href="http://www.digitalglobe.com"/>
            <ns1:ServiceContact>
 <ns1:IndividualName>DigitalGlobe Customer Service</ns1:IndividualName>
            </ns1:ServiceContact>
      </ns1:ServiceProvider>
      <ns1:OperationsMetadata>
            <ns1:Operation name="GetCapabilities">
 <ns1:DCP>
       <ns1:HTTP>
             <ns1:Get
ns2:href="https://services.digitalglobe.com/earthservice/wmtsaccess?CONNECTID=<ConnectI
D>&amp;amp;">

            <ns1:Constraint name="GetEncoding">
 <ns1:AllowedValues>
       <ns1:Value>KVP</ns1:Value>
               </ns1:AllowedValues>
        </ns1:Constraint>
             </ns1:Get>
  </ns1:HTTP>
               </ns1:DCP>
            </ns1:Operation>
            <ns1:Operation name="GetTile">
               <ns1:DCP>
   <ns1:HTTP>
        <ns1:Get
ns2:href="https://services.digitalglobe.com/earthservice/wmtsaccess?CONNECTID=<ConnectI
D>&amp;amp;">
             <ns1:Constraint name="GetEncoding">
   <ns1:AllowedValues>
        <ns1:Value>KVP</ns1:Value>
</ns1:AllowedValues>
         </ns1:Constraint>
    </ns1:Get>
               </ns1:HTTP>
            </ns1:DCP>
      </ns1:Operation>
            <ns1:Operation name="GetFeatureInfo">
<ns1:DCP>
       <ns1:HTTP>
             <ns1:Get
ns2:href="https://services.digitalglobe.com/earthservice/wmtsaccess?CONNECTID=<ConnectI
D>&amp;amp;">
        <ns1:Constraint name="GetEncoding">
<ns1:AllowedValues>
     <ns1:Value>KVP</ns1:Value>
    </ns1:AllowedValues>
   </ns1:Constraint>
 </ns1:Get>
</ns1:HTTP>

…Continued
```

```
Continued…
              </ns1:DCP>
          </ns1:Operation>
      </ns1:OperationsMetadata>
      <ns3:Contents>
            <ns3:Layer>
      </ns3:Contents>
      <ns3:ServiceMetadataURL
ns2:href="https://services.digitalglobe.com/earthservice/wmtsaccess?CONNECTID=
<ConnectID>&amp;SERVICE=WMTS&amp;VERSION=1.0.0&amp;REQUEST=GetCapabilities"/>
</ns3:Capabilities>
       <ns1:Identifier>_null</ns1:Identifier>
 </ns3:Style>
 <ns3:Format>image/png</ns3:Format>
<ns3:TileMatrixSetLink>
  <ns3:TileMatrixSet>EPSG:3395</ns3:TileMatrixSet>
</ns3:TileMatrixSetLink>
<ns3:TileMatrixSetLink>
   <ns3:TileMatrixSet>EPSG:4326</ns3:TileMatrixSet>
 </ns3:TileMatrixSetLink>
 <ns3:TileMatrixSetLink>
```

STEP-2

The client can request a GetTile operation from the WMTS server in order to get the actual imagery tile of a particular tile matrix set in a predefined format. This operation has some parameters in common with WMS GetMap but it has been deliberately simplified.

> Example Request:
>
> https://services.digitalglobe.com/earthservice/wmtsaccess?SERVICE=WMTS&VERSION=1.0.0&STYLE=&REQUEST=GetTile&CONNECTID=<CONNECTID>&LAYER=DigitalGlobe:ImageryTileService&FORMAT=image/jpeg&TileRow=18200&TileCol=27207&TileMatrixSet=EPSG:4326&TileMatrix=EPSG:4326:16&featureProfile=Consumer_Profile
>
> Replace <ConnectID> with the ConnectID provided by DigitalGlobe. Parameters are not required to be in the same order as shown above.

Understanding URL

The parameters required for the GetTile request are shown in Table 3.5. The client provides the following information in a Key-Value Pair (KVP) format, where the “name” field is the key, and the “value” field is the value; the data is supplied in the format “key=value”; for example, “service=WMTS”.

TABLE 3.5 GETTILE REQUEST PARAMETERS

NAME	VALUE	DESCRIPTION
Service*	WMTS	Web Map Tile Service
Version*	1.0.0	Request version
Request*	GetTile	Request name
Layer*	One or more of the available layers, such as: DigitalGlobe:ImageryTileService	The layers available from the Online Catalogs; if more than one layer is requested they are in a comma-separated list. Available layers are advertised in the GetCapabilities response. Refer to Section 3.8 for more information on various WMTS layers.

NAME	VALUE	DESCRIPTION
tileMatrixSet*	String; supported values are: EPSG:4326 EPSG:3857 EPSG:3395	The tileMatrix set to be used to generate the response. A complete list of supported values are found in the GetCapabilities response.
tileMatrix*	String; supported values are: EPSG:4326:0 through EPSG:4326:20 EPSG:3857:0 through EPSG:3857:20	The Tile Matrix identifier of the tileMatrix in the tileMatrixSet requested that has the desired scale denominator that you want to request. 4326 provides tiles where latitude and longitude are treated as X/Y values. 3857 provides tiles in the spherical mercator projection.
tileRow*	Integer	The Row location of the tile in the defined tileMatrixSet.
tileCol*	Integer	The Column location of the tile in the defined tileMatrixSet.
Format*	Mime type of the tile to be returned, for example: image/png image/jpeg	The tile format to return.
connectId*	Character String	A unique 32-digit alphanumeric value assigned by DigitalGlobe (explained under CONNECTID in Section 3.5.3). A valid CONNECTID is mandatory for every request.
featureProfile	The current profiles include: Accuracy_Profile Aerial_CIR_Profile Cloud_Cover_Profile Consumer_Profile Currency_Profile Currency_RGB_Profile Default_Profile True_Currency_Profile	These profiles describe the stacking rules on an account. The user can choose a profile in a request by specifying the featureProfile. The available profiles are advertised in the GetCapabilities response.
exceptions	application/vnd.ogc.se_xml	Format in which exceptions will be reported; if not specified the default is XML.

*Mandatory

Response

The response to a GetTile Request is a map tile, in the requested format and projection. All DGCS-WMTS tiles are 256x256 pixels; no other tile sizes are currently supported.

3.7 Service Exceptions

In the event that a WMTS encounters an error while processing a request or receives an unrecognized request, it will generate an XML document indicating that an error has occurred. An <ExceptionReport> element will contain one or more processing exceptions specified using the <Exception> element. Individual <Exception> elements contain exceptionCode, which specifies the actual exception that occurred.

The following is an example of an exception report. This exception indicates that a parameter value is missing and a parameter value is invalid in the request URL.

```
<?xml version="1.0" encoding="UTF-8"?>
<ExceptionReport xmlns="http://www.opengis.net/ows/1.1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/ows/1.1 owsExceptionReport.xsd"
version="1.0.0" xml:lang="en">
      <Exception exceptionCode="MissingParameterValue" locator="service"/>
      <Exception exceptionCode="InvalidParameterValue" locator="version"/>
</ExceptionReport>
```

3.8 WMTS Layers

TABLE 3.6 WMTS LAYERS

OGC LAYER(S)	DESCRIPTION
ImageryTileService	Tiled imagery for all data layers available to the account, with the default display for a location determined by stacking profile.
CitySphereTileService	Tiled imagery from the Global Basemap City Program. Stacking profiles do not apply.
CountryCoverageTileService	Tiled imagery from the Global Basemap Country Program. Stacking profiles do not apply.

3.9 API Reference

This section provides a list of all possible request parameters for every WMTS operation and detailed information about corresponding responses.

The client should provide the respective information in a KeyValue Pair (KVP) format for every WMS request, where the “name” field is the key and the "value" field is the value. The data is supplied in the format “key=value”; for example, “service=WMTS”.

GetCapabilities

The following table lists all possible request parameters for the GetCapabilities operation of the WMTS server.

TABLE 3.7 UNDERSTANDING URL PARAMETERS FOR WMTS GETCAPABILITIES

PARAMETER NAME	PARAMETER VALUE	DESCRIPTION
CONNECTID*	**<ConnectID>** provided by Digital Globe	Value for this parameter is an unique 32-digit alphanumeric value given by DigitalGlobe (explained under CONNECTID in Section 3.5.3). A valid CONNECTID is mandatory for every request.
SERVICE*	WMTS	Refer to “SERVICE” in Section 3.5.3.
REQUEST*	GetCapabilities	The value for this parameter should always be “**GetCapabilities**” for step-1.
VERSION	1.0.0	Refer to “VERSION” in Section 4.5.3.

* Mandatory parameter

4 Building a Tile Cache

4.1 Introduction

This section describes the requirements and procedure to build a tile cache into the client repository online in real-time. Tile caching is the process by which images are downloaded and saved to a cache for faster retrieval, thus improving performance of client applications. While navigating through AOIs, the client program can search for tiles at respective locations on the local disk without requesting DGCS-WMTS every time. If the tiles are not available on the local disk for the specified region, the request can be sent to DGCS-WMTS GetTile request and the tiles can be saved accordingly.

4.2 Building a Tile Cache from DigitalGlobe WMTS

A client program can communicate with the DGCS WMTS server using the GetTile request for a set of valid Tile Rows and Tile Columns at a given zoom level to build the Tile Cache. The client program has to request **GetCapabilities** from the WMTS server in order to determine what the WMTS server can do and what features the WMTS server can provide. More specifically, in a typical WMTS Tile-Cache client/server interaction, the following steps can be followed:

4.2.1 STEP 1: BUILD TILE CACHE

Identify the Tile Row, Tile Column range for the required AOI and request for GetTile at different zoom levels and required file formats to receive map tile response. Refer to *Introduction to WMTS* on page 12 for more details.

4.2.2 STEP 2: GETTILE REQUEST

An example request for a tile is:

> Name: Imagery
> Sample URL:
> https://services.digitalglobe.com/mapservice/wmtsaccess?SERVICE=WMTS&VERSION=1.0.0&REQUEST=GetTile&LAYER=DigitalGlobe:ImageryTileService&FORMAT=image/jpeg&STYLE=&CONNECTID=<CONNECTID>&TileMatrixSet=EPSG:3857&TileMatrix=EPSG:3857:<ZOOM_LEVEL>&TileRow=<TILE_ROW>&TileCol=<TILE_COLUMN >
>
> Replace <ConnectID> with the ConnectID provided by DigitalGlobe. Replace <TILE_ROW>, <TILE_COLUMN>, and <ZOOM_LEVEL> with the required values. Parameters are not required to be in the same order as shown above.

The above given URL contains Base URL, and parameters as explained in Basic Service Elements on page 14. The key parameter for this request is "**request=GetTile**" which fetches the tiles of WMTS and responds in the form of an image.

The Parameters required for the GetTile request are shown in Table 3.5.

4.2.3 STEP 3: CACHING GETTILE RESPONSE

The response to a Tile Cache Request is a map tile, in the requested format and projection. All DGCS-WMTS tiles are 256x256 pixels; no other tile sizes are currently supported.

In the following GetTile output response image, you may note the values of Tile Rows and Tile Columns at zoom level 16.

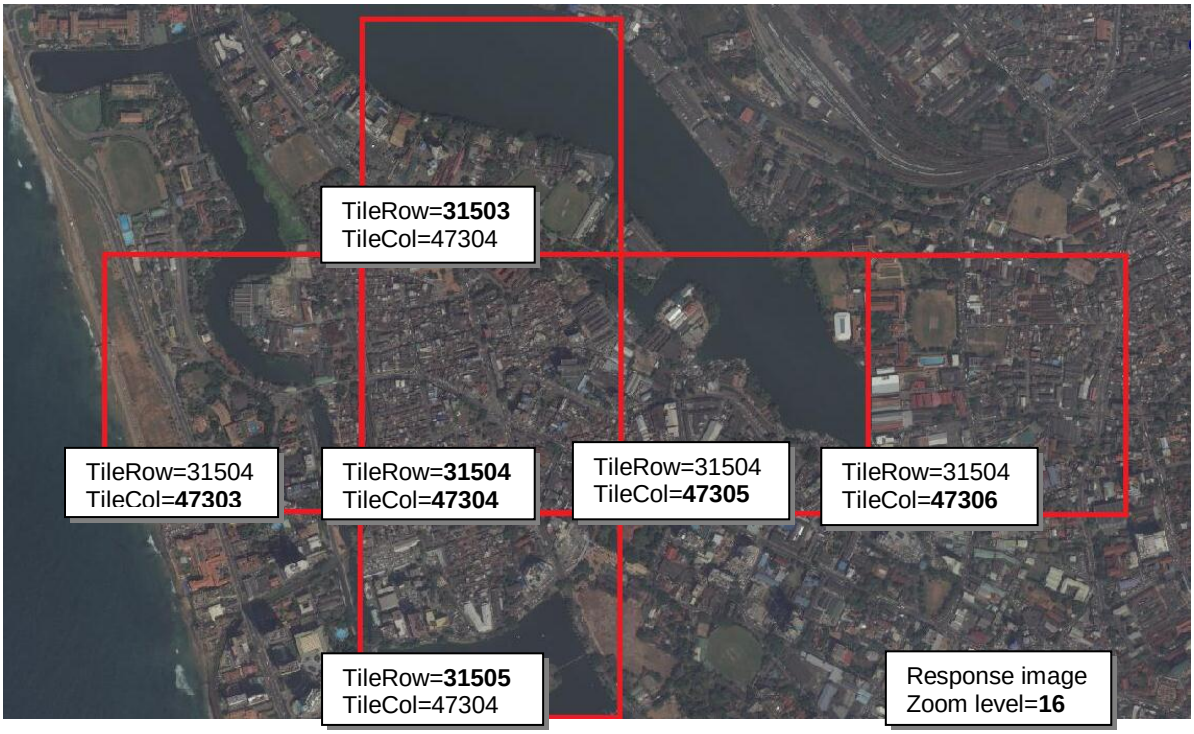

FIGURE 4.1 TILE CACHE RESPONSE IMAGE

On successful building, the tile bytes are received from a series of requests, which can be read through respective file operations API of any programming language and can be saved to local disk/required location. A suggested folder structure to save map tiles locally is given below.

<Zoom Level>
  <Tile Row>
    <Tile Column>.<File Format>
    <Tile Column>.<File Format>
    <Tile Column>.<File Format>

Example:

16
  31504
    47303.jpg
    47304.jpg
    47305.jpg
    47306.jpg

Assumptions

- Tile Cache QA/Error processing must be detected and raised by the user.
- A certain number of threads (simultaneous requests) are allowed to access the system, which is reportable, is provided to the user.
- The customer may choose from metadata in two formats (tab-delimited or XML).
- The tiles X,Y,Z necessary to cover landmass AOI down to zoom 18, should be known by the user.
- It is recommended only going to zoom level 12 for portions of the AOI that are all ocean.
- User should know how to transform the filenames into any caching strategy.
- User would build their ESRI World-file, as needed.

4.3 Updating Tile Cache

To determine the process by which you should update the tile cache, contact your DigitalGlobe representative.

4.4 World-File Generation

If needed by the application, you must generate your own world-file for each tile. The following link describes the contents of a world file: http://en.wikipedia.org/wiki/World_file.

The z, x, and y values are used to translate from indexes to valid Lat/Lon values. The Lat/Lon values are used for lines 5 and 6 of the world-file.

Glossary

AOI

Area of Interest. The area on the Earth that you want to view.

Bilinear Interpolation

Bilinear interpolation uses the value of the four nearest cell centers to determine the value on the output raster. The new value is a weighted average of these four values, adjusted to account for their distance from the center of the output cell. The result is a smoother-looking surface than provided by “nearest neighbor”.

Bicubic Interpolation

Bicubic interpolation combines data points on a two-dimensional grid. This method outputs the smoothest surface of all interpolation methods.

GeoTIFF format

A GeoTIFF file is a TIFF file that is embedded with geographic data tags.

GML

Geography Markup Language. GML is XML code used to express geographical features.

Nearest Neighbor Interpolation

Uses the value of the closest point and disregards all other values, yielding a piecewise-constant interpolant.

OGC

Open GIS Consortium. An international standards organization comprised of commercial, governmental, nonprofit and research organizations. They support geospatial content development as well as data processing and sharing.

OWS

OGC Web Service Common.

Partition

The unit of measure based on the tile zoom level grid for tar file creation for imagery tiles. All tiles and associated metadata for a partition will be tar-compressed into a single file.

UTM

Universal Transverse Mercator Geographic Coordinate System. UTM utilizes a two-dimensional Cartesian system to specify locations on the Earth’s surface.

WCS

Web Coverage Service.

WFS

Web Feature Service.

WMS

Web Map Service.

WMTS

Web Map Tile Service.

Index