Map Library Usage

There are several supported map libraries available to take advantage of NASA's Global Imagery Browse Services (GIBS) via OGC Web Map Tile Service (WMTS), Tiled Web Map Service (TWMS), and KML files. Listed below is a set of map libraries, code snippets, working examples, and screen captures designed to help in building your own user interface to explore NASA's Earth imagery. Alternatively, these guidelines can also help to integrate that imagery into your existing client or to build scripts to retrieve imagery from GIBS. Geographic Information System (GIS) users: go here for instructions.

Code Available in GitHub

In addition to the snippets below, fully-working code examples are available for OpenLayers and Leaflet in the GIBS GitHub project area.

Web-based Libraries

OpenLayers 3

OpenLayers 3 is an open source, web browser-based mapping library written in JavaScript. OpenLayers maps can also be viewed on mobile devices since no plugins are required to use it.

See the Live Examples (including Geographic, Web Mercator, Arctic, and Antarctic, notes on using OpenLayer 3 with GIBS, and clone the GIBS Web Examples from GitHub to tinker with the code. The EOSDIS Worldview tool (source) is based on OpenLayers 3, too.

Leaflet

Leaflet is an open source, web browser-based and mobile-friendly mapping library written in JavaScript.

See the Live Examples (including Geographic, Web Mercator, Arctic, and Antarctic), notes on using Leaflet with GIBS, and clone the GIBS Web Examples from GitHub to tinker with the code. For details on how to configure specific layers available from GIBS, follow the relevant instructions included in the GIBS Layer Configuration Information.

Google Maps

Maps created with the Google Maps API, version 3 can include layers from GIBS. See the GIBS Web Examples for sample code. For details on how to configure specific layers available from GIBS, follow the relevant instructions included in the GIBS Layer Configuration Information .

Bing Maps

Maps created with the Bing Maps AJAX Control, version 7 can include layers from GIBS. See the GIBS Web Examples

for sample code. For details on how to configure specific layers available from GIBS, follow the relevant instructions included in the GIBS Layer Configuration Information .

Google Earth (browser plugin)

Google Earth (Plugin) is a virtual globe that can be embedded on a webpage and manipulated through Javascript. It utilizes the KML interface of TWMS to retrieve data.

Requirements

Google Earth Plugin

Limitations

Google Earth does not have a convenient method of layering maps on top of each other. One can achieve the effect by changing the elevation that the map layers are displayed at, although that sometimes can introduct unwanted artifacts on the display.

Usage

The Javascript uses a callback function to load data from a URL. The list of available products can be found here.

```
var url =
"https://gibs.earthdata.nasa.gov/twms/epsg4326/best/kmlgen.
cgi?layers=MODIS_Terra_CorrectedReflectance_TrueColor&time=
2012-07-12";
google.earth.fetchKml(ge, url, function(object){
 if (!object) {
  // wrap alerts in API callbacks and event handlers
  // in a setTimeout to prevent deadlock in some browsers
  setTimeout(function() {
   displayDialog('Bad or null KML.');
  }, 0);
  return;
 }

 ge.getFeatures().appendChild(object);
 object.setVisibility(true);
 //object.setOpacity(alpha); // use to change transparency
of layer
});
}
```

Screenshots

? Unknown Attachment

GIBS data displayed in the Google Earth Plugin (As used in PO.DAAC's State of the Ocean when "MODIS True Color" layers are selected)

Additional Documentation

Google Earth Plugin API

Desktop Application Libraries

NASA World Wind

World Wind is a multi-platform virtual globe developed by NASA that runs in Java. It can load GIBS data through either the KML interface or the TWMS interface.

Requirements

NASA World Wind 1.3+ or the online JNLP demos

Java 1.6+

Usage (KML)

To run the World Wind client with GIBS KML capabilities, load the "KML Viewer" demo application from either the standalone client or the online demo. To load a specific layer, select File | Open URL... and enter a URL for the layer in this format: *https://<url to kmlgen.cgi>?layers=<layername>&time=<time>* for example, *https://gibs.earthdata.nasa.gov/twms/epsg4326/best/kmlgen.cgi?layers=MODIS_Terra_CorrectedReflectance_TrueColor&time=2012-06-21*

To enable time adjustment on a particular layer, use the following URL parameter: *time=R10/<date>/P1D* for example, *https://gibs.earthdata.nasa.gov/twms/epsg4326/best/kmlgen.cgi?layers=MODIS_Terra_CorrectedReflectance_TrueColor&time=R10/2012-05-26/P1D*

Additional documentation is available here.

Limitations: The name of the layer has to be known prior to entering the URL. The list of available layers can be found here.

Usage (TWMS)

To run the World Wind client with TWMS capabilities, load the WMS Layer Manager demo application from either the

standalone client or the online demo. In the layers menu, add the TWMS endpoint. This will load the list of layers available from TWMS and selecting a layer will allow it to be shown on the globe.

Limitations: World Wind will continue to zoom and try to load images even if they're beyond the depth of the highest resolution tile. When this happens, the images will go blank. In addition, this interface does not allow the selection of variables such as time.

Screenshots

There are no images attached to this page.

Google Earth (Desktop)

See the Google Earth section on the GIBS GIS Usage page.

Mobile App Libraries

ESRI iOS Client Library

The iOS client library is a native library written in Objective-C and can be incorporated into any iOS app for the iPod Touch, iPhone, and iPad. It is built upon ArcGIS's iOS client libraries, which provides a MapView and asynchronous tile loading, coupled with a custom TWMS data loader.

The library comes with a sample client to display TWMS maps and includes basic manipulations such as presenting data from arbitrary TWMS server, selecting a time-value, and map layer reordering.

Requirements

iOS 5.0+

ArcGIS iOS Client Library 2.2.1+ (Download here)

Usage

To use the library, create an AGSMapView object, either programmatically or linked from Interface Builder, and use the following code snippet as an example.

```
TWMSTiledWebMapService *earthTwms =
[[TWMSTiledWebMapService alloc] initWithUrl:[NSURL
URLWithString:@"https://gibs.earthdata.nasa.gov/twms/epsg43
26/best/twms.cgi?request=GetTileService"]];
TWMSTiledMap *aquaMap = [[earthTwms tiledMaps]
valueForKey:@"MODIS AQUA tileset"];
TWMSTiledMapLayer *aquaMapLayer = [[TWMSTiledMapLayer
alloc] initWithTiledMap:aquaMap];
[agsMapView addMapLayer:aquaMapLayer withName:[aquaMap
name]];
```

Screenshots

There are no images attached to this page.

Additional Documentation

The ArcGIS iOS documentation can be obtained here.

The standard iOS documentation can be viewed here.

The iOS library defines 3 classes that are used to load TWMS data:

TWMSTiledWebMapService: Instantiated for each TWMS endpoint.

TWMSTiledMap: Represents an individual map layer in the TWMS.

TWMSTiledMapLayer: A subclass of ArcGIS's map layer to show TWMS data.

TWMSTiledWebMapService

Properties

name: The name of the TWMS as defined in the GetTiledService request.

title: The title of the TWMS as defined in the GetTiledService request.

abstract: The abstract of the TWMS as defined in the GetTiledService request.

tiledMaps: A dictionary where the keys are the name of each map layer and values are the TWMSTiledMap instances.

Methods

(id)initWithUrl:(NSURL)url*: Loads the TWMS data given by the url to instantiate this class.

(NSArray)listAllTiledMapsByName*: Returns a sorted list of map layer names.

(void)setTiledMapKeyValue:(NSString)value forTiledMapKey:(NSString*)key*: Assigns variable keys for all TiledMaps in the TWMS server.

(NSDictionary)tiledMapKeyValues*: Returns a dictionary of all the keys used in all TiledMaps and the value its currently set to.

TWMSTiledMap Properties

name: The name of the layer as defined in the GetTiledService request.

title: The title of the layer as defined in the GetTiledService request.

uuid: A randomly generated unique ID for the layer.

Code Snippets

Create a new TWMSTiledWebMapService using a URL

```
TWMSTiledWebMapService *earthTwms =
[[TWMSTiledWebMapService alloc] initWithUrl:[NSURL
URLWithString:@"https://gibs.earthdata.nasa.gov/twms/epsg43
26/best/twms.cgi?request=GetTileService"]];
```

List all map layers in a TWMS server

```
NSArray *tiledMapNames = [earthTwms
listAllTiledMapsByName];
for(NSString *tiledMapName in tiledMapNames) {
 NSLog(@"%@", tiledMapName);
}
```

Update variables for all TWMS endpoints

```
[earthTwms setTiledMapKeyValue:@"2012-07-12"
forTiledMapKey:@"${time}"];
```

Add, reorder, and remove map layers (Provided by ArcGIS's MapView class)

```
AGSMapView *mapView = <the map view>;
TWMSTiledMap *tiledMap = <obtained from
TWMSTiledWebMapService>;
TWMSTiledMapLayer *tiledMapLayer = [[TWMSTiledMapLayer
alloc] initWithTiledMap:tiledMap];

[mapView addLayer:tiledMapLayer withName:[tiledMap uuid]];
[mapView insertMapLayer:tiledMapLayer withName:[tiledMap
uuid] atIndex:<index>];
[mapView removeLayerWithName:[tiledMap uuid]];
```

Set map layer transparency (Provided by iOS UIView's transparency settings)

```
UIView *view = [[[mapView mapLayerViews]
objectForKey:[tiledMap uuid]];
[view setAlpha:<transparency value>];
```

Script-level Access

GDAL (Basics)

The Geospatial Data Abstraction Library (GDAL) WMS driver supports several internal 'minidrivers' that allow access to different web mapping services. Each of these services may support a different set of options in the *Service* block. Documentation for these minidrivers can be found here on the GDAL website. Two of these minidrivers in particular can be used by users to download GIBS imagery programmatically. They are the Tile Map Specification (TMS) and the OnEarth Tiled WMS (TiledWMS) minidrivers. Examples for both of these minidrivers will be included below.

Requirements

GDAL version 1.9.1 or greater with cURL support enabled. To check if cURL is enabled for GDAL, type "gdalinfo --format WMS". If cURL is enabled you will see information about the WMS format, if not, you will get an error message and you will need to reconfigure GDAL to support cURL.

Sample Execution #1 - "TMS" Driver Configuration File Input

Create a local service description XML file and invoke gdal_translate. In this example, GIBS_Aqua_MODIS_true.xml is used to generate a true color JPEG image from Aqua MODIS of a smoke plume from western fires over the central United States. The contents of GIBS_Aqua_MODIS_true.xml would be:

```
<GDAL_WMS>
    <Service name="TMS">

<ServerUrl>https://gibs.earthdata.nasa.gov/wmts/epsg4326/be
st/MODIS_Aqua_CorrectedReflectance_TrueColor/default/2013-0
8-21/250m/${z}/${y}/${x}.jpg</ServerUrl>
    </Service>
    <DataWindow>
        <UpperLeftX>-180.0</UpperLeftX>
        <UpperLeftY>90</UpperLeftY>
        <LowerRightX>396.0</LowerRightX>
        <LowerRightY>-198</LowerRightY>
        <TileLevel>8</TileLevel>
        <TileCountX>2</TileCountX>
        <TileCountY>1</TileCountY>
        <YOrigin>top</YOrigin>
    </DataWindow>
    <Projection>EPSG:4326</Projection>
    <BlockSizeX>512</BlockSizeX>
    <BlockSizeY>512</BlockSizeY>
    <BandsCount>3</BandsCount>
</GDAL_WMS>
```

```
gdal_translate -of GTiff -outsize 1200 1000 -projwin -105
42 -93 32 GIBS_Aqua_MODIS_true.xml GreatPlainsSmoke1.tif
gdal_translate -of JPEG GreatPlainsSmoke1.tif
GreatPlainsSmoke1.jpg
```

In very limited testing, our experience has been that better image quality is obtained by using the GeoTIFF output format from the GDAL WMS, then converting this to other formats, if desired, using a second gdal_translate command, or other programs such as ImageMagick convert.

Sample Execution #2 - "TMS" Driver Command Line Input

Invoke gdal_translate with the content of the local service description XML file embedded into the command. This approach is useful for automated scripting to download various layers, dates, etc. To generate the same image as above:

```
gdal_translate -of GTiff -outsize 1200 1000 -projwin -105
42 -93 32 '<GDAL_WMS><Service
name="TMS"><ServerUrl>https://gibs.earthdata.nasa.gov/wmts/
epsg4326/best/MODIS_Aqua_CorrectedReflectance_TrueColor/def
ault/2013-08-21/250m/${z}/${y}/${x}.jpg</ServerUrl></Servic
e><DataWindow><UpperLeftX>-180.0</UpperLeftX><UpperLeftY>90
</UpperLeftY><LowerRightX>396.0</LowerRightX><LowerRightY>-
198</LowerRightY><TileLevel>8</TileLevel><TileCountX>2</Til
eCountX><TileCountY>1</TileCountY><YOrigin>top</YOrigin></D
ataWindow><Projection>EPSG:4326</Projection><BlockSizeX>512
</BlockSizeX><BlockSizeY>512</BlockSizeY><BandsCount>3</Ban
dsCount></GDAL_WMS>' GreatPlainsSmoke2.tif
gdal_translate -of JPEG GreatPlainsSmoke2.tif
GreatPlainsSmoke2.jpg
```

For both options, the information needed to create the local service description XML can be found in the large table on the GIBS Available Imagery Products page.

Items that may need to be changed include:

1) <ServerUrl>https://gibs.earthdata.nasa.gov/wmts/**epsgcode**/best/**layer_name**/default/**date**/**resolution**/${z}/${y}/${x}.**format**</ServerUrl>

layer_name - use "Layer Name on Server" from Products page. Note that only one layer can be retrieved per gdal_translate call.

date - use the desired date in YYYY-MM-DD format

epsgcode - use "epsg" followed by the appropriate EPSG code from Footnote 4 on the GIBS Available Imagery Products page

resolution - use the "Resolution" in the format ####m from the "Imagery Resolution" column on the GIBS Available Imagery Products page, or GoogleMapsCompatible_Level# resolution from table below

format - use jpg or png extension based on the "Format"

2) **Bounding box** - use -180.0, 90, 396.0, -198 for Geographic projection and -4194304, 4194304, 4194304, -4194304 for the Polar projections. This is the bounding box of the topmost tile, which matches the bounding box of the actual imagery for Polar but not for Geographic.

3) **<TileLevel>** - select from the table below based on "Resolution" and "Projection". Note that the TileLevel in the table below is the maximum for that resolution. You can specify a lower value of TileLevel to force GDAL to use coarser resolution input tiles. This can be used to speed up projection, for example during development and testing.

Resolution	TileLevel (Geographic)	TileLevel (Polar)	Resolution (Web Mercator)	TileLevel (Web Mercato
2km	5	2	GoogleMapsCompatible_Level6	6
1km	6	3	GoogleMapsCompatible_Level7	7
500m	7	4	GoogleMapsCompatible_Level8	8
250m	8	5	GoogleMapsCompatible_Level9	9
125m	9	-	GoogleMapsCompatible_Level10	10
62.5m	10	-	GoogleMapsCompatible_Level11	11
31.25m	11	-	GoogleMapsCompatible_Level12	12
15.625m	12	-	GoogleMapsCompatible_Level13	13

4) **<TileCountX><TileCountY>** - use 2, 1 for Geographic projection and 2, 2 for Polar projections

5) **<Projection>** - use the appropriate EPSG code from Footnote 4 on the GIBS Available Imagery Products page.

6) **<Bands>** - use 3 for .jpg (except use 1 for ASTER_GDEM_Greyscale_Shaded_Relief) and 4 for .png

Note that the values for <YOrigin>, <BlockSizeX>, and <BlockSizeY> are constant for all GIBS layers.

More information on GDAL_WMS can be found here (note the "TMS" section).

Sample Execution #3 - "TiledWMS" Driver Configuration File Input

The TiledWMS GDAL minidriver relies on a simplified set of XML configuration from the user, pulling all other information from the TWMS *WMS_Tile_Service* document at runtime. The following example requests the same image as was requested in Sample Execution #1, but you will notice how much simpler the XML configuration is. The primary difference is that requests through TWMS utilize a *TiledGroupName* value instead of the *WMTS Identifier* utilized by WMTS requests. In almost all cases the *TiledGroupName* can be generated by replacing all underscores in a WMTS layer's *Identifier* with spaces and appending " tileset". For example, a WMTS Layer *Ident ifier* of SMAP_L1_Passive_Faraday_Rotation_Aft" becomes the TWMS *TiledGroupName* "SMAP L1 Passive Faraday Rotation Aft tileset". Unfortunately there are a few exceptions, which include the sample layer used below.

```
<GDAL_WMS>
  <Service name="TiledWMS">

<ServerUrl>https://gibs.earthdata.nasa.gov/twms/epsg4326/be
st/twms.cgi?</ServerUrl>
    <TiledGroupName>MODIS AQUA tileset</TiledGroupName>
    <Change key="${time}">2013-08-21</Change>
  </Service>
</GDAL_WMS>
```

```
gdal_translate -of GTiff -outsize 1200 1000 -projwin -105
42 -93 32 GIBS_Aqua_MODIS_true.xml GreatPlainsSmoke1.tif
gdal_translate -of JPEG GreatPlainsSmoke1.tif
GreatPlainsSmoke1.jpg
```

Sample Execution #4 - "TiledWMS" Driver Command Line Input

This example invokes gdal_translate with the content of the TileWMS local service description XML file embedded as a command line argument. This approach is useful for automated scripting to download various layers, dates, etc. To generate the same image as above, run the following:

```
gdal_translate -of GTiff -outsize 1200 1000 -projwin -105
42 -93 32 '<GDAL_WMS><Service
name="TiledWMS"><ServerUrl>https://gibs.earthdata.nasa.gov/
twms/epsg4326/best/twms.cgi?</ServerUrl><TiledGroupName>MOD
IS AQUA tileset</TiledGroupName><Change
key="${time}">2013-08-21</Change></Service></GDAL_WMS>'
GreatPlainsSmoke2.tif
gdal_translate -of JPEG GreatPlainsSmoke2.tif
GreatPlainsSmoke2.jpg
```

GDAL (Advanced)

Configuration File Input w/ Transparent PNG

In this example, GIBS_OMI_AI.xml is used to generate a transparent PNG image from the OMI Aerosol Index over the same region as the first two examples. The "Layer Name on Server" and EPSG/resolution in <ServerURL>, the <TileLevel>, and the <BandsCount> have changed from the first examples. The contents of

GIBS_OMI_AI.xml would be:

```
<GDAL_WMS>
    <Service name="TMS">

<ServerUrl>https://gibs.earthdata.nasa.gov/wmts/epsg4326/be
st/OMI_Aerosol_Index/default/2013-08-21/2km/${z}/${y}/${x}.
png</ServerUrl>
    </Service>
    <DataWindow>
        <UpperLeftX>-180.0</UpperLeftX>
        <UpperLeftY>90</UpperLeftY>
        <LowerRightX>396.0</LowerRightX>
        <LowerRightY>-198</LowerRightY>
        <TileLevel>5</TileLevel>
        <TileCountX>2</TileCountX>
        <TileCountY>1</TileCountY>
        <YOrigin>top</YOrigin>
    </DataWindow>
    <Projection>EPSG:4326</Projection>
    <BlockSizeX>512</BlockSizeX>
    <BlockSizeY>512</BlockSizeY>
    <BandsCount>4</BandsCount>
</GDAL_WMS>
```

```
gdal_translate -of GTiff -outsize 1200 1000 -projwin -105
42 -93 32 GIBS_OMI_AI.xml GreatPlainsSmokeAI.tif
gdal_translate -of PNG GreatPlainsSmokeAI.tif
GreatPlainsSmokeAI.png
```

Polar Imagery

In this example, GIBS_Terra_MODIS_Arctic.xml is used to generate a true color JPEG image from Terra MODIS of a phytoplankton bloom in the Barents Sea. The "Layer Name on Server" and EPSG/resolution in <Serverurl>, the <DataWindow> elements, and the <Projection> have changed from the first examples. The contents of GIBS_Terra_MODIS_Arctic.xml would be:

```
<GDAL_WMS>
    <Service name="TMS">

<ServerUrl>https://gibs.earthdata.nasa.gov/wmts/epsg3413/be
st/MODIS_Terra_CorrectedReflectance_TrueColor/default/2013-
08-01/250m/${z}/${y}/${x}.jpg</ServerUrl>
    </Service>
    <DataWindow>
        <UpperLeftX>-4194304</UpperLeftX>
        <UpperLeftY>4194304</UpperLeftY>
        <LowerRightX>4194304</LowerRightX>
        <LowerRightY>-4194304</LowerRightY>
        <TileLevel>5</TileLevel>
        <TileCountX>2</TileCountX>
        <TileCountY>2</TileCountY>
        <YOrigin>top</YOrigin>
    </DataWindow>
    <Projection>EPSG:3413</Projection>
    <BlockSizeX>512</BlockSizeX>
    <BlockSizeY>512</BlockSizeY>
    <BandsCount>3</BandsCount>
</GDAL_WMS>
```

GeoTIFF files created by GDAL using the default tags for the GIBS polar projections do not work correctly in some GIS programs. So we recommend specifying the output projection in PROJ4 format:

```
gdal_translate -of GTiff -outsize 980 940 -a_srs
"+proj=stere +lat_0=90 +lat_ts=70 +lon_0=-45 +k=1 +x_0=0
+y_0=0 +ellps=WGS84 +datum=WGS84 +units=m +no_defs"
-projwin 1520000 240000 2500000 -700000
GIBS_Terra_MODIS_Arctic.xml BarentsSea.tif
gdal_translate -of JPEG BarentsSea.tif BarentsSea.jpg
```

The projection string for the GIBS Antarctic projection is "+proj=stere +lat_0=-90 +lat_ts=-71 +lon_0=0 +k=1 +x_0=0 +y_0=0 +ellps=WGS84 +datum=WGS84 +units=m +no_defs". Here are the links for the Arctic or Antarctic PROJ4 definitions. Some other workarounds are:

1) Use a ".aux.xml" file with the ".tif" file. Download the sample Arctic or Antarctic XML file and rename it to match the GeoTIFF file.

```
gdal_translate -of GTiff -outsize 980 940 -projwin 1520000
240000 2500000 -700000 GIBS_Terra_MODIS_Arctic.xml
BarentsSea.tif
"copy" GIBS_Arctic_3413.tif.aux.xml "to"
BarentsSea.tif.aux.xml
gdal_translate -of JPEG BarentsSea.tif BarentsSea.jpg
```

2) Use JPEG format instead. This will create a .jpg.aux.xml file in addition to the .jpg file.

```
gdal_translate -of JPEG -outsize 980 940 -projwin 1520000
240000 2500000 -700000 GIBS_Terra_MODIS_Arctic.xml
BarentsSea.jpg
```

Polar Layer w/ Reprojection

In this example, GIBS_Aqua_MODIS_Arctic.xml is used to generate a true color JPEG image from Aqua MODIS of the Beaufort Sea reprojected to have 145W down. The "Layer Name on Server" and EPSG/resolution in <Serverurl>, the <DataWindow> elements, and the <Projection> have changed from the first examples. The contents of GIBS_Aqua_MODIS_Arctic.xml would be:

```
<GDAL_WMS>
    <Service name="TMS">

<ServerUrl>https://gibs.earthdata.nasa.gov/wmts/epsg3413/be
st/MODIS_Aqua_CorrectedReflectance_TrueColor/default/2013-0
6-20/250m/${z}/${y}/${x}.jpg</ServerUrl>
    </Service>
    <DataWindow>
        <UpperLeftX>-4194304</UpperLeftX>
        <UpperLeftY>4194304</UpperLeftY>
        <LowerRightX>4194304</LowerRightX>
        <LowerRightY>-4194304</LowerRightY>
        <TileLevel>5</TileLevel>
        <TileCountX>2</TileCountX>
        <TileCountY>2</TileCountY>
        <YOrigin>top</YOrigin>
    </DataWindow>
    <Projection>EPSG:3413</Projection>
    <BlockSizeX>512</BlockSizeX>
    <BlockSizeY>512</BlockSizeY>
    <BandsCount>3</BandsCount>
</GDAL_WMS>
```

```
gdalwarp -t_srs '+proj=stere +lat_0=90 +lat_ts=70
+lon_0=-145 +k=1 +x_0=0 +y_0=0 +datum=WGS84 +units=m
+no_defs' -te -1600000 -2500000 1200000 -400000 -tr 1000
1000 GIBS_Aqua_MODIS_Arctic.xml Beaufort_neg145down_1km.tif
gdal_translate -of JPEG Beaufort_neg145down_1km.tif
Beaufort_neg145down_1km.jpg
```

Reprojecting GIBS Geographic layer into Polar Stereographic

Only a subset of layers are available from GIBS in the polar projections. This is an example of how to project a geographic layer into a polar projection. The contents of GIBS_Terra_Cloud_Fraction.xml would be:

```
<GDAL_WMS>
    <Service name="TMS">

<ServerUrl>https://gibs.earthdata.nasa.gov/wmts/epsg4326/be
st/MODIS_Terra_Cloud_Fraction_Day/default/2017-05-16/2km/${
z}/${y}/${x}.png</ServerUrl>
    </Service>
    <DataWindow>
        <UpperLeftX>-180.0</UpperLeftX>
        <UpperLeftY>90</UpperLeftY>
        <LowerRightX>396.0</LowerRightX>
        <LowerRightY>-198</LowerRightY>
        <TileLevel>5</TileLevel>
        <TileCountX>2</TileCountX>
        <TileCountY>1</TileCountY>
        <YOrigin>top</YOrigin>
    </DataWindow>
    <Projection>EPSG:4326</Projection>
    <BlockSizeX>512</BlockSizeX>
    <BlockSizeY>512</BlockSizeY>
    <BandsCount>4</BandsCount>
    <ZeroBlockHttpCodes>204,404,400</ZeroBlockHttpCodes>
</GDAL_WMS>
```

```
gdalwarp -wo SOURCE_EXTRA=100 -wo SAMPLE_GRID=YES -of GTiff
-t_srs "+proj=stere +lat_0=90 +lat_ts=70 +lon_0=-45 +k=1
+x_0=0 +y_0=0 +ellps=WGS84 +datum=WGS84 +units=m +no_defs"
-ts 4096 4096 -te -4194304 -4194304 4194304 4194304
GIBS_Terra_Cloud_Fraction.xml
Terra_Cloud_Fraction_Arctic.tif
```

The "-wo SOURCE_EXTRA=100 -wo SAMPLE_GRID=YES" arguments are needed to ensure that the output region is completely filled. The "-wo SOURCE_EXTRA=100" caused GDAL to request tiles outside of input bounding region. Normally GDAL would halt when it receives these errors so the <ZeroBlockHttpCodes> includes the 400 error to allow GDAL to continue. Note that this command may take a long time to complete, especially at higher input and output resolutions.