WORKING DRAFT

Implementing Moving Animations in Web Mapping Service

Scott Hamilton[1], Ryan Boyles[2], Fred Bernal[2],Matt McMahon[2], Eric LaMar[2]

[1]*Galaxy Global Corp., Fairmont, WV*
[2]*The Institute for Scientific Research, Inc., Fairmont, WV*
Email of corresponding author: shamilton@galaxyglobal.com (Scott Hamilton)

Abstract

The purpose of this paper is to discuss the details of implementing animations within the current Web Mapping Server (WMS) Specification. The discussion will include the details of the specification that allow for the implementation and conclusions as to how the specification can be extended to allow for more fluid implementation of animations in general. The objective is to extend the WMS specification to include animations that move based on geographical coordinates and time.

1. Introduction

WMS is a web-based tool that produces and serves digital maps. In the past, analyzing and creating maps was a slow and labor intensive process. Now with Global Positioning Systems and other digital methods of gathering geographical data, digital mapping has become an important component in several fields. The WMS specification is a web-based protocol that allows communication of map-style data between a server and a client.

WMS, and digital imaging in general, has a wide range of applications. Law enforcement and national defense are two major areas of focus. For example, if the government had a map server of digital surveillance photos or satellite images of an area, those images could be accessed by law enforcement agencies all over the world, helping in the war on terrorism. There are several other practical uses of WMS, such as weather forecasting, tourism, and transportation. Google, for example, has a browser called Google Earth that accesses several WMS servers. It is a 3-D browser that contains layers including restaurants, road maps, recreational locations, landmarks, as well as satellite imagery. Their browser is proprietary and prevents you from adding your own WMS server to its list, but it is a perfect example of the power of WMS.

Work began with WMS through installing the University of Minnesota's custom implementation of a WMS server called MapServer. UMN packaged MapServer into a Linux-based operating system called HOST GIS (Global Information Systems). HOST GIS houses all of the code (written in C and C++) that UMN wrote to implement their map server. UMN MapServer has two major components it utilizes in order to produce a map. The first is called a map file. A map file contains information on multiple layers of images, and a "bounding box" of latitude and longitude coordinates you set for your images. The second, an ArcView World File (WLD) file, is a file that is written for each and every image on the UMN map server. The WLD file contains specific latitude and longitude coordinates as well as the pixel size of the images. These two types of files have to work hand in hand to make images appear properly.

However, UMN's MapServer isn't fully compliant with the WMS spec., which means that in order to do animations, one must more or less write an implementation of map server and not use UMN's. One major issue that was encountered is that UMN did not implement the time extents in a robust manner, limiting the ability to change a map image over time. Obviously, this is a necessity in implementing any kind of animation.

As a result, an implementation of WMS was developed in house to address the issues with UMN's implementation. The implementation has many limitations and does not implement the entire specification, only those pieces necessary to demonstrate the ability of the specification to serve animated maps which transverse both time and space was implemented. Moving animations were very successfully served to NASA's World Wind (http://worldwind.arc.nasa.gov/) WMS client . To the knowledge of the team, World Wind is the only WMS-enabled 3-D client to support animations.

2. Specification Details

The WMS Specification consists of two basic system calls. The first is GetCapabilities which returns an XML document adhering to a standard set of parameters. This XML document gives details about map layers, the servers capabilities (i.e. the image formats, projections, and geographic bounds of the server.) See Appendix A for a sample capabilities XML document with additional comments. Below is a sample GetCapabilities request:

http://hostgis.isr.us/cgi-bin/scottwms/wms.pl?request=GetCapabilities&service=WMS&version=1.1.1

The next part of the WMS specification defines the format for requesting a map(image) from a WMS server based on the capabilities file. The best way to define the GetMap request is to give an example request and break down the details.

http://hostgis.isr.us/cgi-bin/scottwms/wms.pl?service=WMS&version=1.1.1&request=GetMap&layers=Charley&format=image/png&width=512&height=512&time=2004-08-12T15:55Z&srs=EPSG:4326&bbox=-91.00,11.8,-68,34.50&styles=&transparent=TRUE

Every WMS request has certain components regardless of whether you are requesting a map or a capabilities documents. See table below.

Server Name	*http://hostgis.isr.us*
Script Path	*/cgi-bin/scottwms/wms.pl*
Service Name	*Service=WMS* (OGC defines other services ie WFS, etc)
Version	*Version=1.1.1* (Versions are not entirely backward compatible so the version number is required to determine the proper format for the request.
Request Type	*request=GetMap* or *request=GetCapabilites* (in Version 1.1.1)

The parameters one is interested in for creating animations come from the GetMap request type and are outlined below.

Layer Name	*layer=Charley* (Any layer from the XML capabilities file)
Image format	*format=image/png* (Any of several formats, the severs capabilities file will give a list of all formats a server is capable of delivering. Our sample server is limited to png.)
Image width	*width=512* (can be any size, the output image will be this number of pixels wide).
Image height	*height=512* (can be any size, the output image will be this number of pixels high).
Image time (Optional needed for animations, not required for a compliant sever implementation.)	*time=2004-08-12T15:55Z* (the time stamp of a particular image, time is one of many user defined constraints that can be placed on a map image and usually ties to a database field that contains the image for a particular time, these custom constraints are defined as a part of the capabilities document and according to the specification must be optional in order to be compliant.)
Image Projection	*srs=EPSG:4326* (This can be any EPSG Projection code, the WMS specification requires all compliant map servers to provide imagery in at least EPSG:4326 so that was chosen for our example.)
Geographic Bounding Box	*bbox=-91.00,11.8,-68,34.50* (The geographical coordinates of the corners of the image upper-left, lower-right specified in the Projection model from the SRS field.) This is the total area in which the client wants the data projected into. It can not be larger than the bounding box specified by the capabilities document, but it can be smaller, in which case only a portion of the defined image will be displayed.

Layer Name	*layer=Charley* (Any layer from the XML capabilities file)
Output styles	*styles=* Optionally specify a format for the output document.
Transparency of the image	*transparent=TRUE* (Specifies whether the output image should contain transparency information.)

There is one other aspect of the specification that allows one to provide animations that move geographically over time. The specification serves geo-referenced images. The Geospatial Data Abstraction Library (GDAL) is a translator library for raster geospatial data formats that is released under an X/MIT style Open Source license. As a library, it presents a single abstract data model to the calling application for all supported formats. The related OpenGIS Simple Features Reference Implementation (OGR) library (which lives within the GDAL source tree) provides a similar capability for simple features vector data. GDAL and associated libraries can be found at: http://www.gdal.org.

The methods withing GDAL allows one to specify exact coordinates for an image and therefore gives one the ability to place the image for a specified time, within its own bounding box, within the bounding box requested by the client.

3. Implementation Details

Before going into a detailed breakdown of how to implement moving animations let us talk a little about how animations work in general, then discuss the subtle changes that must occur to move the animation over a geographical path with time.

The first experimentation with animations began by exploring the NASA World Wind Project. World Wind is a client containing custom scripts to provide pseudo-animations by displaying a series of still imagery. In essence, the WMS server does not send animations to World Wind, or any other client for that matter, but a series of time stamped imagery. The WMS specification does not provide a method of serving animations, as its primary focus is mapping with still imagery, however it does provide a key element necessary for animations: the ability to associate a time with an image. The specification also allows for individual images within a map layer to have various geo-referenced data, this allows for the pseudo-animation to move geographically over time.

First take a look at a simple animation. For example a volcano before, during, and after an eruption. Images obtained from NASA's Scientific Visualization Server (http://svs.gsfc.nasa.gov).

Mt. St. Helens before, during and after eruption.

In order to animate this event, one must place the images on the server and provide a geographic bounding box, and a series of time stamps within a layer. This will give an XML capabilities document with the follow layer item within it.

<Layer>
<Name>Mt_St_Helens</Name>
<Title>Mount St. Helens Before, During, and After (1024x1024 Animation)</Title>
<Abstract>Mount St. Helens erupted on May 18, 1980, devastating more than 150 square miles of forest in southwestern Washington state. This animation shows Landsat images of the Mount St. Helens area in 1973, 1983, and 2000, illustrating the destruction and regrowth of the forest. The 1983 image clearly shows the new crater on the northern slope where the eruption occurred, the rivers and lakes covered with ash, and the regions of deforestation. The 2000 image, taken twenty years after the eruption, still shows the changed crater, but much of the devastated area is covered by new vegetation growth.</Abstract>
<LatLonBoundingBox minx="-122.36011" miny="46.122115" maxx="-121.97916" maxy="46.384547" />
<BoundingBox SRS="EPSG:4326" minx="-122.36011" miny="46.122115" maxx="-121.97916" maxy="46.384547" />
<Dimension name="time" units="ISO8601"/>
*<Extent name="time" default="09-25-2000">**09-15-1973,05-22-1983,09-25-2000**</Extent>*
</Layer>

The client will then know that the following three map requests will get the three frames of the animation.

http://hostgis.isr.us/cgi-bin/scottwms/wms.pl?service=WMS&version=1.1.1&request=GetMap&layers=Mt_St_Helens&format=image/png&width=512&height=512&time=09-15-1973&srs=EPSG:4326&bbox=-122.36011,46.122115,-121.97916,46.384547&styles=&transparent=TRUE

http://hostgis.isr.us/cgi-bin/scottwms/wms.pl?service=WMS&version=1.1.1&request=GetMap&layers=Mt_St_Helens&format=image/png&width=512&height=512&time=05-22-1983&srs=EPSG:4326&bbox=-122.36011,46.122115,-121.97916,46.384547&styles=&transparent=TRUE

http://hostgis.isr.us/cgi-bin/scottwms/wms.pl?service=WMS&version=1.1.1&request=GetMap&layers=Mt_St_Helens&format=image/png&width=512&height=512&time=09-25-2000&srs=EPSG:4326&bbox=-122.36011,46.122115,-121.97916,46.384547&styles=&transparent=TRUE

It is now up the the client application to play the frames in sequence and in effect create the animation. As one can see, this is a pseudo-animation in the fact that WMS does not actually send an animation to the client, but rather a series of still images. The result, of course, will be an animation.

Huricane Charley Imagery

Now for a sample of a moving animation. Say one has imagery of a hurricane, for simplicity three frames over a period of three days. Images obtained from NASA's Scientific Visualization Server (http://svs.gsfc.nasa.gov).
In order to animate this event, one must place the images on the server and provide a geographic bounding box for each image, and a series of time stamps within a layer. The server will then re-sample the images creating an image that covers the complete bounds of the animation. This is done by mapping the original image over the larger transparent image. This however creates a limitation in that only Portable Network Graphics (png) , Graphic Interchange Format (gif), or Tag(ged) Image File Format (TIFF) images may be used without causing an opaque black background on the animation. This does not cause a server to fall out of specification, but does create issues for some client software.

In order to have the animation move over time, one must set the geographical coordinates of the imagery. This is done using a World coordinates (WLD) file to geo-rectify the imagery. This could be done in any number of ways including a database field. However, this implementation uses the WLD files specified within the GDAL library. The table below shows the WLD files used for each image above to correctly locate them along with a description of each field in the file.

Meaning of Number	***2004-08-11.wld***	***2004-08-12.wld***	***2004-08-13.wld***
x scaling factor	0.014013	0.01542	0.01539
x rotation term.	0.00	0.00	0.00
Y rotation term.	0.00	0.00	0.00
Y scaling factor	-0.01054	-0.01142	-0.01626
Upper Left Longitude	-83.166	-88.8844	-90.515327
Upper Left Latitude	23.39	26.8436	34.26223

This will give an XML capabilities document with the follow layer item within it.

```
<Layer>
    <Name>Charley</Name>
    <Title>Hurricane Charley Animation</Title>
    <LatLonBoundingBox minx="-90.5253" miny="12.6028" maxx="-68.8210" maxy="34.2722" />
    <BoundingBox SRS="EPSG:4326"  minx="-90.5253" miny="12.6028" maxx="-68.8210" maxy="34.2722" />
```

<Dimension name="time" units="ISO8601"/>
*<Extent name="time" default="2004-08-11Z" nearestValue="0">**2004-08-11T18:15Z,2004-08-12T15:55Z,2004-08-13T16:35Z**</Extent>*
</Layer>

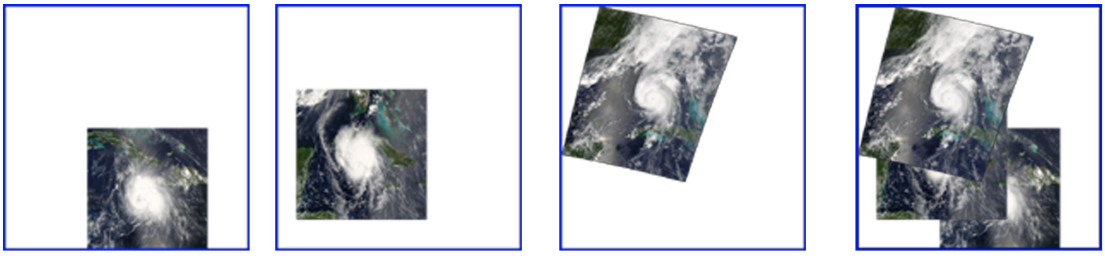
The client will then know that the following three map requests will get the three frames of the animation.

Http://hostgis.isr.us/cgi-bin/scottwms/wms.pl?service=WMS&version=1.1.1&request=GetMap&layers=Charley&format=image/png&width=512&height=512&**time=2004-08-11T18:15Z**&srs=EPSG:4326&bbox=-90.5253,12.6028,-68.8210,34.2722&styles=&transparent=TRUE

http://hostgis.isr.us/cgi-bin/scottwms/wms.pl?service=WMS&version=1.1.1&request=GetMap&layers=Charley&format=image/png&width=512&height=512&**time=2004-08-12T15:55Z**&srs=EPSG:4326&bbox=-90.5253,12.6028,-68.8210,34.2722&styles=&transparent=TRUE

http://hostgis.isr.us/cgi-bin/scottwms/wms.pl?service=WMS&version=1.1.1&request=GetMap&layers=Charley&format=image/png&width=512&height=512&**time=2004-08-13T16:35Z**&srs=EPSG:4326&bbox=-90.5253,12.6028,-68.8210,34.2722&styles=&transparent=TRUE

Take notice the requests are almost identical to the requests for the still imagery with the exception of the layer name. This is because the movement of the imagery is done on the server side and the client always receives an image that fills the entire geographical bounding box from the request. The image will have transparent edges around the true image to locate the image in the correct geographical location. This transparent image is generated by the server at the time of the request. The three images below are what will be returned from the server. The forth image shows them overlaid to indicate the motion in the animation.



Images from the server shown with a border around the transparent space.

Once again, it is up the the client application to play the frames in sequence and in effect create the animation. It is clear from this discussion that an extension to the specification for animations would greatly benefit those people working with time sensitive data such as weather forecasting and flight tracking.

There is one more major issue with this method of a moving animation. It requires that the imagery be re-sampled to create the animation frame, this nearly always degrades the quality of the data. The lack of a method for serving moving imagery without degrading the data quality shows a great need for extending the specification.

4. Test Plan

Perform the following Conformance tests from Annex A of the Open Geospatial Constortium (OGC) Web Map Service (WMS) Implementation (OGC 04-024) Document Version 1.3 (or the version you are attempting conformance). Detailed below are the test cases for each section that were used to test the implementation.
A.1.2 Basic WMS Server
A.1.2.1 Version negotiation.
http://hostgis.isr.us/cgi-bin/scottwms/wms.pl?service=WMS&request=GetCapabilities

http://hostgis.isr.us/cgi-bin/scottwms/wms.pl?service=WMS&request=getcapabilities

http://hostgis.isr.us/cgi-bin/scottwms/wms.pl?service=WMS&request=GetCapabilities&version=1.1.1

Should all return a valid capabilities XML document. See Annex A for an example.

http://hostgis.isr.us/cgi-bin/scottwms/wms.pl?request=GetCapabilities

Should return an error message in our implementation an image stating No Service Requested. (Service=WMS is a required parameter on all requests.)

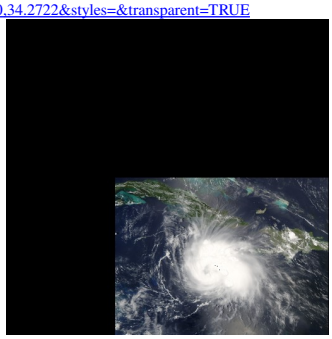
A1.2.2 Request Parameter rules.
This on is a tricky to test, basically you should get a valid http protocol response back from your server even with an incorrect call to the server. Just test a series of valid and invalid urls. Since Apache is used in this implementation, this is covered by the webserver software and not our code.

A.1.2.3 GetCapabilites Response
Same test and results as A.1.2.1

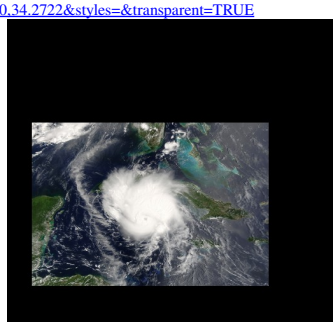
A.1.2.4 GetMap Response
Verify that a valid map is returned for several test case urls. The ones listed below should return the images below them. The black portion of the images represent the transparent layer to show the offset of the image within the bounding box.

These images are the animated sequence using the default bounding box.

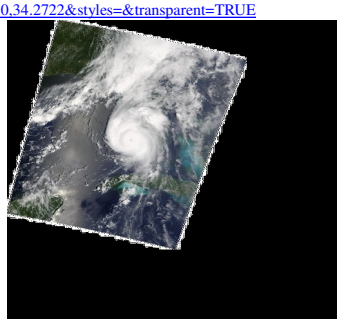
http://hostgis.isr.us/cgi-bin/scottwms/wms.pl?service=WMS&version=1.1.1&request=GetMap&layers=Charley&format=image/png&width=512&height=512&time=2004-08-11T18:15Z&srs=EPSG:4326&bbox=-90.5253,12.6028,-68.8210,34.2722&styles=&transparent=TRUE



http://hostgis.isr.us/cgi-bin/scottwms/wms.pl?service=WMS&version=1.1.1&request=GetMap&layers=Charley&format=image/png&width=512&height=512&time=2004-08-12T15:55Z&srs=EPSG:4326&bbox=-90.5253,12.6028,-68.8210,34.2722&styles=&transparent=TRUE

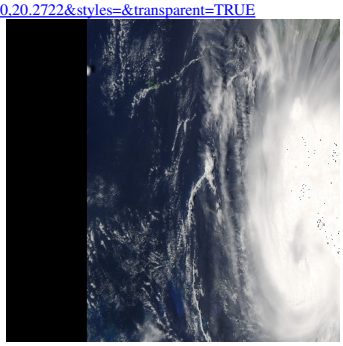


http://hostgis.isr.us/cgi-bin/scottwms/wms.pl?service=WMS&version=1.1.1&request=GetMap&layers=Charley&format=image/png&width=512&height=512&time=2004-08-13T16:35Z&srs=EPSG:4326&bbox=-90.5253,12.6028,-68.8210,34.2722&styles=&transparent=TRUE

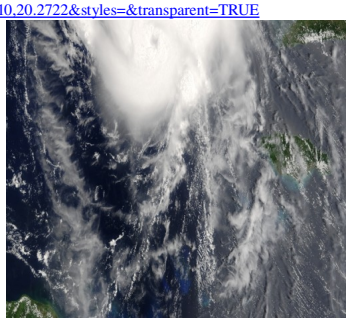


These images are the animated sequence using a bounding box with the northern bound moved south 14 degrees. And the Eastern bound moved East 5 degrees. You will notice that the images look distorted. This is normal in that a square image was requested for a rectangular bounding box.

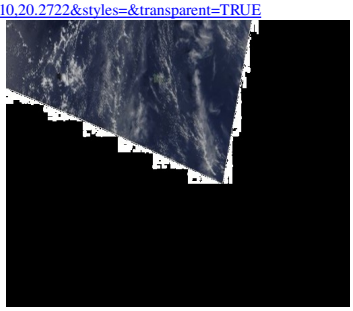
http://hostgis.isr.us/cgi-bin/scottwms/wms.pl?service=WMS&version=1.1.1&request=GetMap&layers=Charley&format=image/png&width=512&height=512&time=2004-08-11T18:15Z&srs=EPSG:4326&bbox=-85.5253,15.6028,-75.8210,20.2722&styles=&transparent=TRUE



http://hostgis.isr.us/cgi-bin/scottwms/wms.pl?service=WMS&version=1.1.1&request=GetMap&layers=Charley&format=image/png&width=512&height=512&time=2004-08-12T15:55Z&srs=EPSG:4326&bbox=-85.5253,15.6028,-75.8210,20.2722&styles=&transparent=TRUE



http://hostgis.isr.us/cgi-bin/scottwms/wms.pl?service=WMS&version=1.1.1&request=GetMap&layers=Charley&format=image/png&width=512&height=512&time=2004-08-13T16:35Z&srs=EPSG:4326&bbox=-85.5253,15.6028,-75.8210,20.2722&styles=&transparent=TRUE



Further verification was done by using World Wind to get the imagery with various bounding boxes and it was verified that the images were correctly located geographically after the scaling and shifting.

5. Specification Extension Recommendations

From the research it appears there are three general ways in which the specification could be extended to provide animations more effectively. There is a major issue with all the suggested extension models in that it could take years to get the specification extended to include any of the recommended extensions. The methods of extending the specification are detailed in the following sections. (5.1) Adding a return type of frame. (5.2) Returning multiple frames. (5.3) Returning a timestamped series of frames as an animation.

5.1 Add a return type of frame.

Instead of simply returning an image from the server, return a multi-part Multipurpose Internet Mail Extensions (MIME) type document that contains XML describing the bounding box and timing parameters for the returned frame, as well as the embedded image in the requested format. This would allow a moving animation to specify the exact coordinates of a given frame rather than having the server do image manipulation to achieve the animations.

5.1.1 Pros to the extension model.

This is a fairly easy extension to implement in that multi-part MIME is directly support by most web browsers and does not go outside the http specification. It will return a single document that contains the original image (no image degradation) and the exact geographical coordinates for the image (more exact placement). This will move all image processing to the client an allow for a lighter weight server.

This model falls within the existing specification to a certain extent in that the specification currently gathers the bounding box information for each layer based on data in the capabilities document. This extension could improve the performance of the specification by transferring some of the layer description information into the frame definition allowing for a more streamlined capabilities document.

5.1.2 Cons to the extension model.

Embedding the image file in a multi-part MIME type transfers the image in a text format rather than binary and will increase the file size dramatically. There may be a better method of encoding the multi-part MIME that will eliminate this issue.

This method is does not solve issues with high latency networks. It still requires the client to request each frame rather than requesting an entire series of frames in a single request.

This method could create issues for some of the simple web based browsers in that they may or may not support multi-part MIME.

5.1.3 Implementation of the model.

An implementation of the model was written in Perl extending a standard WMS implementation. A GetFrame request type was implemented which takes the same parameters as GetMap, only returns a bounding box parameter as a single line of XML code. The model may not be fully compliant with the XML standards, but close enough for proof of concept. The returned document looks like the following:

Content-Type: multipart/mixed; boundary=ISRSCOTTWMSVER01010101

--ISRSCOTTWMSVER01010101
Content-Type: text/xml; charset=ISO-8859-1

<BoundingBox SRS="EPSG:4326" minx=" 32.4012600 " miny=" -89.5012600" maxx=" 29.8207800 " maxy="
-87.0871000" />

Content-Type: image/png

‰PNG (The binary image data)

--ISRSCOTTWMSVER01010101--

The client application must then split the multi-part MIME into two parts, the XML document, and the image file. In the proof of concept code, a modified NASA World Wind client was used. The cleint processed the XML portion of the data in memory and stored the raw image file for processing.

The proof of concept code showed that this model worked very well under most circumstances. It allows one to shift a single image and display it properly located regardless of the requested bounding box. The proof of concept code needs to be extended to crop images that fall on the edges of the requested bounding box before it can be considered a complete implementation.

There was a substantial server performance improvement implementing in this manner, but the animations were slower on the client do to the rendering model within World Wind. It will require some extra work on the client side to see the same level of performance as with server side image processing.

It was discovered that the data size increase was minor and was completely offset in the test case by the limited image processing required by the server. The multi-part MIME seems to be a good solution for single images, or frames in a small animation.

Before moving on to implement the second model, some tests cases were ran by requesting a large 240 frame animation of Hurricane Katrina's landfall event. The high resolution imagery and sheer size of the animation created major problems for both the client, and the server due to network latency. The animation took over 30 minutes to download, proving the need for additional compression of multi-frame animations.

As can be seen from the included screen shots, this method greatly enhanced the image quality. Both images returned are 512x512, but the one on the left is scaled down within a transparent image losing some quality, the one on the right looks stretched, but will be corrected by the WMS client when geographic rectification occurs. The one on the right will clearly be of a higher quality when displayed by the client.

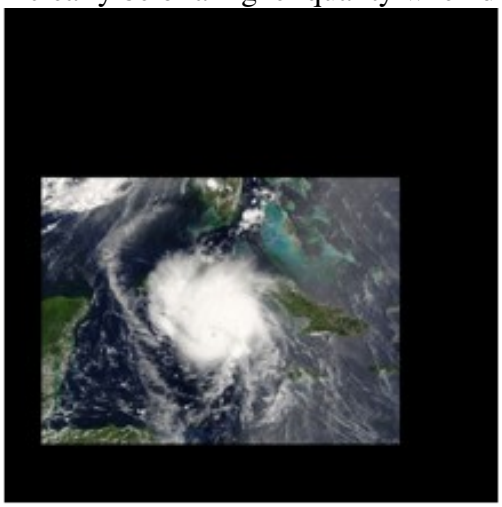


Image from GetMap Request

Image from GetFrame Request

The following URLs were used to receive the above images.

http://hostgis.isr.us/cgi-bin/scottwms/wms.pl?service=WMS&version=1.1.1&request=GetMap&layers=Charley&format=image/png&width=512&height=512&time=2004-08-12T15:55Z&srs=EPSG:4326&bbox=-90.5253,12.6028,-68.8210,34.2722&styles=&transparent=TRUE

http://hostgis.isr.us/cgi-bin/scottwms/wms.pl?service=WMS&version=1.1.1&request=GetFrame&layers=Charley&format=image/png&width=512&height=512&time=2004-08-12T15:55Z&srs=EPSG:4326&bbox=-90.5253,12.6028,-68.8210,34.2722&styles=&transparent=TRUE

5.2 Add an optional GetFrames request to the specification.

Instead of returning a single image, GetFrames would return a series of images with different bounding box parameters. This would be an extension of the frame type specified in section 5.1. This would allow the client to properly locate the returned set of imagery and playback an animation with a single request.

5.2.1 Pros to the extension model.

This model includes the pros from the model discussed in section 5.1. In addition to these pros, this model helps issues with high-latency networks. The ability to request multiple frames with a single request limits the communications from client to server decreasing the network latency because the bulk of traffic will be from server to client.

A big advantage to this model is it does not limit the ability of the client to request only certain frames of an animation. This could allow the client to download any portion of an animation for viewing.

The animation will be returned to the client as a single stream simplifying the communications model tremendously over the current model and model 5.1.

Returned as a multi-part MIME type, the resulting stream can be viewed in most modern web browsers allowing users without specific client software to view the frames of the animation.

5.2.2 Cons to the extension model.

This extension model has the same cons as the previous model with the additional con of creating a very complex MIME return type resulting in a more complicated client side interpretation of the data. This implementation would require a higher performance client machine.

5.2.3 Implementation of the model.

An implementation of the model was written in Perl extending a standard WMS implementation. A GetFrames request type was implemented which takes the same parameters as GetMap, only it returns a sequence of XML bounding box parameters followed by an image file, followed by then next frame in the same format. This model implementation may not be fully compliant with the XML standards, but close enough for proof of concept. The returned document looks like the following:

Content-Type: multipart/mixed; boundary=ISRSCOTTWMSVER01010101

```
--ISRSCOTTWMSVER01010101
Content-Type: text/xml; charset=ISO-8859-1

<BoundingBox SRS="EPSG:4326" minx="  32.4012600 " miny="   -89.5012600" maxx="  29.8207800 " maxy="
-87.0871000" />

Content-Type: image/png

‰PNG   (The binary image data)

--ISRSCOTTWMSVER01010101
Content-Type: text/xml; charset=ISO-8859-1

<BoundingBox SRS="EPSG:4326" minx="  32.4012600 " miny="   -89.5012600" maxx="  29.8207800 " maxy="
-87.0871000" />

Content-Type: image/png

‰PNG   (The binary image data)
//sequence repeats until all frames are served.
--ISRSCOTTWMSVER01010101--
```

The client application must then split the multi-part MIME into many parts: the XML documents, and image files. In the proof of concept code, a modified NASA World Wind client was used. The XML portion of the data was processed in memory and the raw image files were stored for processing. Each image/XML file pair was treated as if they were a separate download request in the client proving that the streaming method works, without having to recode a lot of the display engine.

The proof of concept code showed that this model worked very well under most circumstances. It allowed the server to send multiple images and display them properly located regardless of the requested bounding box. The proof of concept code needs to be extended to crop images that fall on the edges of the requested bounding box before it can be considered a complete implementation.

There was substantial performance improvement over both the original implementation and method 5.1 at the server implementing in this manner. However, the animations appear slower on the client do to the rendering model within World Wind. It will require some extra work on the client side to see the same level of performance as with server side image processing.

It was discovered that the data size increase returning multiple frames in a single stream was substantial and created a significant delay before the client could display the first frame. However, subsequent frames displayed more rapidly and network traffic was reduced.

Before moving on to implement the third model some tests were ran by requesting a large 240 frame animation of Hurricane Katrina's landfall event. The high resolution imagery and sheer size of the animation created major problems for both the client, and the server due to the size of the data returned. The animation still took around 15 minutes to download and in that time frame the client is waiting for the first frame, proving the need for additional compression of multi-frame animations.

It was discovered that the client performance improvement over model 5.1 and 5.2 to be negligible from a computer standpoint model 5.2 being slightly faster, but as an end user, model 5.1 appears faster since there is less time between a request, and viewing the first frame of the animation.

In conclusion, when it comes to high-resolution, multiple frame animations, neither suggested model 5.1 or 5.2 is very effective, and leaves the user waiting for imagery to transfer from the server.

5.3 Add an optional GetMovie request to the specification.

Instead of returning an image, or series of images GetMovie would return an avi, animated gif, or other movie format along with an XML document defining the time stamp, and georeferencing data locating each frame to allow the client to move the animation along the path. These would be embedded in a multi-part MIME type.

5.3.1 Pros to the extension model.

Much smaller size to the returned data set than returning single frames. Movies are much smaller in size due to the nature of video compression algorithms. Each frame rather than being stored as a full image is stored as a variance from the prior image. This compression would be very effective in hurricane models as the change is minor between individual frames over time giving the ability to serve higher resolution imagery at a much higher rate. Streaming the movie would allow for an apparently faster download of the animation to the client.

5.3.2 Cons to the extension model.

Individual frames or a subset of the frames would not be readily available to the client unless the server generated the movie on the fly. Generating the movie on the fly would create large processor overhead on the server, as well as create a delay in rendering the movie.

There is not a movie format available aside from animated gif that will allow for transparency. This prevents the client from easily overlaying the movie onto a map or other image and comparing the movie to the underlying imagery. This can be overcome by extracting the frames from the movie as individual images with transparency settings.

5.3.3 Implementation of the model.

An implementation of the model was written in Perl extending a standard WMS implementation. A GetMovie request type was implemented which takes the same parameters as GetMap, excluding the bounding box parameters as they are returned from the server. The GetMovie request in our proof of concept code returns a multi-part MIME type. The returned data includes an XML document that contains the bounding box and time stamp parameters for each frame of the animation. The XML document is followed by an animation sequence of images compressed using MSMPEG4 encoding. The animation sequence is encoded by the server and can only be returned to the client as a whole in our implementation. However, there are tools available to allow individual frames of the animation to be sent from the server upon request, as well as the ability to encode the frames on the fly at the server side if so desired.

The implementation proved that both of the Cons to the implementation model can be easily overcome. The inability to serve individual frames is overcome by software tools that allows one to send individual frames of a compressed video file over a stream, making a subset of the frames available to the client without the need to generate the movie on the fly at the server level. The inability to serve transparent images is overcome by setting a particular image color as transparent, at the client level, in the animation and using tools of the client to convert the compressed animation into individual frames using any image format the client desires.

Under the hood, this implementation works by encoding the imagery as a MSPEG4 movie. This movie is appended to a multi-part MIME document containing information about each frame. This is sent to the client. Any web browser can receive the mutli-part MIME and has the capabilities of splitting the XML from the movie and playing the movie. However, the standard web browser does not know how to handle the geo-referencing information so the animation will not move in space.

The implementation when given the url:
http://hostgis.isr.us/cgi-bin/scottwms/wms.pl?service=WMS&version=1.1.1&request=GetMovie&layers=ErinTrack
returns a Multipart MIME type document similar to the following:
Content-Type: multipart/mixed; boundary=ISRSCOTTWMSVER01010101

--ISRSCOTTWMSVER01010101

Content-Type: text/xml; charset=ISO-8859-1

<?xml version='1.0' encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE WMT_MS_Capabilites SYSTEM "http://schemas.opengeospatial/wms/1.1.1/WMS_MS_Capabilities.dtd">
<LAYER>
<BoundingBox SRS="epsg:4326" minx="21.3" miny="-74.2" maxx="23.3" maxy="-72.2" time="1995-07-31T00:00Z"/>
<BoundingBox SRS="epsg:4326" minx="21.6" miny="-74.6" maxx="23.6" maxy="-72.6" time="1995-07-31T06:00Z"/>
<BoundingBox SRS="epsg:4326" minx="21.8" miny="-74.9" maxx="23.8" maxy="-72.9" time="1995-07-31T12:00Z"/>
<BoundingBox SRS="epsg:4326" minx="22.2" miny="-75.3" maxx="24.2" maxy="-73.3" time="1995-07-31T18:00Z"/>
<BoundingBox SRS="epsg:4326" minx="22.6" miny="-75.9" maxx="24.6" maxy="-73.9" time="1995-08-01T00:00Z"/>
<BoundingBox SRS="epsg:4326" minx="23.3" miny="-76.7" maxx="26.3" maxy="-74.7" time="1995-08-01T06:00Z"/>
<BoundingBox SRS="epsg:4326" minx="24.5" miny="-77.3" maxx="26.5" maxy="-75.3" time="1995-08-01T12:00Z"/>
<BoundingBox SRS="epsg:4326" minx="25.3" miny="-78.7" maxx="27.3" maxy="-76.7" time="1995-08-01T18:00Z"/>
<BoundingBox SRS="epsg:4326" minx="25.9" miny="-80.0" maxx="27.9" maxy="-78.0" time="1995-08-02T00:00Z"/>
<BoundingBox SRS="epsg:4326" minx="26.7" miny="-81.4" maxx="28.7" maxy="-79.4" time="1995-08-02T06:00Z"/>
<BoundingBox SRS="epsg:4326" minx="27.2" miny="-82.9" maxx="29.2" maxy="-80.9" time="1995-08-02T12:00Z"/>
<BoundingBox SRS="epsg:4326" minx="27.6" miny="-84.4" maxx="29.6" maxy="-82.4" time="1995-08-02T18:00Z"/>
<BoundingBox SRS="epsg:4326" minx="27.8" miny="-85.7" maxx="29.8" maxy="-83.7" time="1995-08-03T00:00Z"/>
<BoundingBox SRS="epsg:4326" minx="28.3" miny="-86.7" maxx="30.3" maxy="-84.7" time="1995-08-03T06:00Z"/>
<BoundingBox SRS="epsg:4326" minx="28.8" miny="-87.6" maxx="30.8" maxy="-85.6" time="1995-08-03T12:00Z"/>
<BoundingBox SRS="epsg:4326" minx="29.6" miny="-88.5" maxx="31.6" maxy="-86.5" time="1995-08-03T18:00Z"/>
<BoundingBox SRS="epsg:4326" minx="30.4" miny="-89.5" maxx="32.4" maxy="-87.5" time="1995-08-04T00:00Z"/>
<BoundingBox SRS="epsg:4326" minx="31.3" miny="-90.1" maxx="33.3" maxy="-88.1" time="1995-08-04T06:00Z"/>
<BoundingBox SRS="epsg:4326" minx="32.2" miny="-90.7" maxx="34.2" maxy="-88.7" time="1995-08-04T12:00Z"/>
<BoundingBox SRS="epsg:4326" minx="33.1" miny="-91.2" maxx="35.1" maxy="-89.2" time="1995-08-04T18:00Z"/>
<BoundingBox SRS="epsg:4326" minx="33.8" miny="-91.2" maxx="35.8" maxy="-89.2" time="1995-08-05T00:00Z"/>
<BoundingBox SRS="epsg:4326" minx="34.4" miny="-91.1" maxx="36.4" maxy="-89.1" time="1995-08-05T06:00Z"/>
<BoundingBox SRS="epsg:4326" minx="35.3" miny="-90.8" maxx="37.3" maxy="-88.8" time="1995-08-05T12:00Z"/>
<BoundingBox SRS="epsg:4326" minx="36.5" miny="-89.8" maxx="38.5" maxy="-87.8" time="1995-08-05T18:00Z"/>
<BoundingBox SRS="epsg:4326" minx="37.4" miny="-87.8" maxx="39.4" maxy="-85.8" time="1995-08-06T00:00Z"/>
<BoundingBox SRS="epsg:4326" minx="37.7" miny="-85.9" maxx="39.7" maxy="-83.9" time="1995-08-06T06:00Z"/>
<BoundingBox SRS="epsg:4326" minx="37.8" miny="-83.0" maxx="39.8" maxy="-81.0" time="1995-08-06T12:00Z"/>
</LAYER>
--ISRSCOTTWMSVER01010101
Content-Type: video/avi
The encoded movie data.
--ISRSCOTTWMSVER01010101--

Just as a proof of concept, mplayer was used on the client side to split the movie into PNG image files, and the existing engine within World Wind displayed the animation frames. The performance increase was surprising, the test animations of 24 frames was loading about 10 frames per minute using all other implementation models discussed and had a total download size of just over 12MB. Using this model, the first frame was displayed within 10 seconds of the request nearly immediately followed by the rest. The download size dropped from just over 12MB to just over 1MB.

It was decided to perform the same test using the Hurricane Katrina Landfall event of 240 frames that took nearly 30 minutes using the other methods and downloaded over 240MB of data. Using our new model the download took less than 2 minutes and has a size of only 5.6MB. The processing time had a little to be desired in that extracting 240 frames took a substantial amount of time on the client, however it was significantly less than download time required with the other models.

After much testing it was decided that even with the extra client overhead involved using a movie encoding technique to transfer the frames of an animation is a very viable solution to the animation issues within WMS. The ideal model would be one in which the raw movie file could be played and moved across a geographic path, without decompressing the frames, but even converting the frames to images at the client level showed profound performance improvement over sending the individual frames across the network.

It is important to add that the testing was performed using a 100MB local area network to transmit the data and these performance numbers would show even greater benefit to the model if tested across a slower link.

6. Conclusions

Implementation Model	***Pros***	***Cons***
5.1 Get Frame	• Easy to implement. • Follows current specification to a certain degree. • Could streamline the capabilities document.	• MIME types increase the data transfer size. • Still has issues with high latency networks. • Could break the ability to view the data with a web browser.
5.2 Get Frames	• Easy to implement. • Follows current specification to a certain degree. • Could streamline the capabilities document. • Eliminates some issues with high latency networks. • Does not limit the ability to request single frames. • A much simplified communications model.	• MIME types increase the data transfer size. • Still has issues with high latency networks. • Could break the ability to view the data with a web browser. • More complex MIME type will require a higher performance client to view the data. • Definitely creates issues viewing the data in a web browser. • Large delay before first frame displays.
5.3 Get Movie	• Much smaller data transfer. • Great improvement on high latency networks. • Simplified communications model. • Streaming the video feed will cause rapid appearance of first frame. • Video compression techniques work well with hurricane models.	• Individual frames would not be readily available to the client for viewing or download. • Generating a movie on demand will require a high end server. • Implementation of the client is much more complex. • Transparency of the data becomes an issue with most movie formats.

In conclusion, the best overall solution would be to combine methods for the most flexibility in serving animations:

- Using the Get Movie method for a large sequence of images, having the movie generated prior to the request.
- Using the Get Frames method for sequences containing a portion of an event.

7. Extras

Source code for the server implementation is included in Appendix A. The server requirements are outlined in Appendix B. Appendix C contains a sample wms.xml file. Appendix D contains the difference files to make World Wind work using this server. The World Wind modification will require that MPLAYER be installed in the c:/mplayer directory on the client machine. MPLAYER can be obtained from http://www.mplayerhq.hu

Appendix A Server source code.

```
#!/usr/bin/perl

#
# env.pl
#
# This script dumps the environment variables in HTML format

use strict;
use warnings;
use CGI;
my $query = new CGI;
my $debug = "0";
#my $hostname = "http://10.0.2.209/mywms/wms.pl";
#my $hostname = $ENV{DOCUMENT_ROOT}.$ENV{SCRIPT_NAME};
#my $hostname = "http://hostgis.isr.us/cgi-bin/scottwms/wms.pl";
#print $query->header("text/xml");
my $key1="SCRIPT_NAME";
my $key2="HTTP_HOST";
my $foo;
my $hostname = "http://".$ENV{$key2}.$ENV{$key1};
#my $hostname = "http://hostgis.isr.us".$ENV{$key1};
my $filename;
my $getframe="0";
#print $hostname;
my @names = $query->param; #Get all the parameter names from the url.
# To get values from the name use $value = $query->param($name);
if ($query->param("service") =~ m/^WMS$/i)
{
 foreach my $name (@names) {
        if ($name =~ m/^request$/i) {
                if ($query->param($name) =~ m/^GetCapabilities$/i) {
                        goto GETCAP;
                }
                else
                {
        if ($query->param($name) =~ m/^GetMap$/i) {
                        goto GETMAP;
        }
        else
        {
          if ($query->param($name) =~ m/^GetFrames$/i) {
            goto GETFRAMES;
          }
          else
          {
            if ($query->param($name) =~ m/^GetFrame$/i) {
              $getframe="1";
              goto GETMAP;
            }
            else
            {
              if ($query->param($name) =~ m/^GetMovie$/i) {
                goto GETMOVIE;
              }
              else
              {
                $filename="/maps/data/errors/invrequest.png";
```

```
                goto END;
               }
              }
             }
            }
           }
          }
 }
 $filename="/maps/data/errors/norequest.png";
 goto END;
}
$filename="/maps/data/errors/noservice.png";
goto END;

# Get Capabilities Code
GETCAP:print $query->header("text/xml");
open(xml, "wms.xml");
my @xml_data=<xml>;
close(xml);
foreach my $item (@xml_data) {
        $item=~ s/\$hostname/$hostname/g; #replaces $hostname in xml with value of $hostname
        print $item;
}
goto END2;

GETMAP:
if ($debug eq "1") {print $query->header("text/html");}
my $layer;
my $time;
my $bbox;
my $width;
my $height;
my $format;
my $srs;
my $styles;
my $transparent;
my $version="0";
foreach my $name (@names) {
        if ($name =~ m/^layers$/i) {
                $layer=$query->param($name);
        }
        if ($name =~ m/^format$/i) {
                $format=$query->param($name);
        }
        if ($name =~ m/^width$/i) {
                $width=int($query->param($name));
        }
        if ($name =~ m/^height$/i) {
                $height=int($query->param($name));
        }
        if ($name =~ m/^time$/i) {
                $time=$query->param($name);
        }
        if ($name =~ m/^srs$/i) {
                $srs=$query->param($name);
        }
        if ($name =~ m/^bbox$/i) {
                $bbox=$query->param($name);
        }
        if ($name =~ m/^styles$/i) {
```

```
                        $styles=$query->param($name);
                }
                if ($name =~ m/^transparent$/i) {
                        $transparent=$query->param($name);
                }
        if ($name =~ m/^version$/i) {
                $version=$query->param($name);
        }
}
# Version Number is required.
if ($version eq "0") {
  $filename="/maps/data/errors/nover.png";
  goto END;
}
# Version Number must equal the implementation number.
if ($version ne "1.1.1") {
  $filename="/maps/data/errors/invver.png";
  goto END;
}
# Format is required.
if ($format eq "") {
#  print "No Format";
  $filename="/maps/data/errors/invrequest.png";
  goto END;
}
# Format must be image/png for my server.
if ($format =~ m/^image\/png$/i) {
  #ok
} else {
#  print "Wrong Format";
  $filename="/maps/data/errors/invrequest.png";
  goto END;
}
# Width required,
if ($width eq "") {
#  print "No width";
  $filename="/maps/data/errors/invrequest.png";
  goto END;
}
# Height required.
if ($height eq "") {
#  print "No Height";
  $filename="/maps/data/errors/invrequest.png";
  goto END;
}
# Bbox required.
if ($bbox eq "") {
#  print "No bbox";
  $filename="/maps/data/errors/invrequest.png";
  goto END;
}
# SRS required.
if ($srs eq "") {
#  print "No SRS";
  $filename="/maps/data/errors/invrequest.png";
  goto END;
}
# SRS must be EPSG:4326 for my server.
if ($srs ne "EPSG:4326") {
#  print "Wrong SRS";
```

```
  $filename="/maps/data/errors/invrequest.png";
  goto END;
}
if ($layer eq "") {
#  print "No Layers";
  $filename="/maps/data/errors/invrequest.png";
  goto END;
}
if ($time eq "") {$time="01"}
#my $wldfilename="c:\\isoimages\\WMS\\MyOwnServer\\data\\".$layer."\\".$time.".wld";
my $wldfilename="/maps/data/".$layer."/".$time.".wld";
#my $filename="c:\\isoimages\\WMS\\MyOwnServer\\data\\".$layer."\\".$time.".jpg";
$filename="/maps/data/".$layer."/".$time.".jpg";
if ($debug eq "1") {print $query->header("text/html");}
#print $filename;
my $bminlon;
my $bmaxlon;
my $bminlat;
my $bmaxlat;
my $d = `date`;
$d =~ s/\//_/g;
$d =~ s/^\s+//;
$d =~ s/\s+$//;
$d =~ s/\s/_/g;
$d =~ s/:/_/g;
my $tmpimage="/maps/tmp/".$layer.$d.".png";
my $bkgtmpimage="/maps/tmp/bkg".$layer.$d.".png";
my $fgtmpimage="/maps/tmp/fg".$layer.$d.".png";
#if ($debug eq "1") {print $tmpimage."<BR>";
($bminlon,$bminlat,$bmaxlon,$bmaxlat)=split(/,/,$bbox);
if (open(WLD, $wldfilename)){
  my $latscale=<WLD>;
  my $rot1=<WLD>;
  my $rot2=<WLD>;
  my $lonscale=<WLD>;
  my $ullon=<WLD>;
  my $ullat=<WLD>;
  my $lrlat;
  my $lrlon;
  my $blatscale;
  my $blonscale;
  my $gdalinfo=`gdalinfo \"$filename\"`;
  my @gdallines=split(/\n/,$gdalinfo);
  foreach my $gdalline (@gdallines){
    if ($gdalline =~ m/Upper Left/){
      $gdalline =~ s/Upper Left//;
      $gdalline =~ s/\(//;
      $gdalline =~ s/\)//;
      ($ullon,$ullat)=split(/,/,$gdalline);
      if ($debug eq "1") {print "Found UpperLeft Lat:".$ullat."<BR>";}
      if ($debug eq "1") {print "Found UpperLeft Lon:".$ullon."<BR>";}
    }
    if ($gdalline =~ m/Lower Right/){
      $gdalline =~ s/Lower Right//;
      $gdalline =~ s/\(//;
      $gdalline =~ s/\)//;
      ($lrlon,$lrlat)=split(/,/,$gdalline);
      if ($debug eq "1") {print "Found LowerRight Lat:".$lrlat."<BR>";}
      if ($debug eq "1") {print "Found LowerRight Lon:".$lrlon."<BR>";}
    }
```

```
}
if ($debug eq "1") {print "Found Boudning Box Upper Left Lat:".$bmaxlat."<BR>";}
if ($debug eq "1") {print "Found Bounding Box Upper Left Lon:".$bminlon."<BR>";}
if ($debug eq "1") {print "Found Boudning Box Lower Right Lat:".$bminlat."<BR>";}
if ($debug eq "1") {print "Found Bounding Box Lower Right Lon:".$bmaxlon."<BR>";}
if ($debug eq "1") {print "Image xscale:".$latscale."<BR>";}
if ($debug eq "1") {print "Image yscale:".$lonscale."<BR>";}
$blatscale=($bmaxlat-$bminlat)/$width;
$blonscale=($bmaxlon-$bminlon)/$height;
if ($debug eq "1") {print "BBox xscale:".$blatscale."<BR>";}
if ($debug eq "1") {print "BBox yscale:".$blonscale."<BR>";}
my $imglat=($ullat-$lrlat);
my $imglon=abs($ullon-$lrlon);
if ($debug eq "1") {print "Image Width Lat:".$imglat."<BR>";}
if ($debug eq "1") {print "Image Height Lon:".$imglon."<BR>";}
$imglat=int($imglat/$blatscale);
$imglon=int($imglon/$blonscale);
if ($debug eq "1") {print "Image Width Lat:".$imglat."<BR>";}
if ($debug eq "1") {print "Image Height Lon:".$imglon."<BR>";}
#my $imgoffsetlon=(-$bminlon+$ullon);
#my $imgoffsetlat=($bmaxlat-$ullat);
#print "Image Offset Lat:".$imgoffsetlat."<BR>";
#print "Image Offset Lon:".$imgoffsetlon."<BR>";
#$imgoffsetlat=$imgoffsetlat/$blatscale;
#$imgoffsetlon=$imgoffsetlon/$blonscale;
#if ($imgoffsetlat>0) {$imgoffsetlat="+".$imgoffsetlat;}
#if ($imgoffsetlon>0) {$imgoffsetlon="+".$imgoffsetlon;}
#print "Image Offset Lat:".$imgoffsetlat."<BR>";
#print "Image Offset Lon:".$imgoffsetlon."<BR>";
#Define new corners it lat and lon degrees.
my $ulcornerlat;
my $ulcornerlon;
my $lrcornerlat;
my $lrcornerlon;
if ($ullon<$bminlon) {
  $ulcornerlon=$ullon;
} else {
  $ulcornerlon=$bminlon;
}
if ($ullat>$bmaxlat) {
  $ulcornerlat=$ullat;
} else {
  $ulcornerlat=$bmaxlat;
}
if ($lrlon>$bmaxlon) {
  $lrcornerlon=$lrlon;
} else {
  $lrcornerlon=$bmaxlon;
}
if ($lrlat<$bminlat) {
  $lrcornerlat=$lrlat;
} else {
  $lrcornerlat=$bminlat;
}
#if ($debug eq "1") {print "Temp Image ulcornerlon:".$ulcornerlon."<BR>";}
#if ($debug eq "1") {print "Temp Image ulcornerlat:".$ulcornerlat."<BR>";}
#if ($debug eq "1") {print "Temp Image lrcornerlon:".$lrcornerlon."<BR>";}
#if ($debug eq "1") {print "Temp Image lrcornerlat:".$lrcornerlat."<BR>";}
# Define the crop region for the image to be served.
#my $high=abs(($lrcornerlat-$ulcornerlat)/$blatscale);
```

```
#my $wide=abs(($ulcornerlon-$lrcornerlon)/$blonscale);
#my $cropoffsetlat=($ulcornerlat-$bmaxlat)/$blatscale;
#my $cropoffsetlon=(-$lrcornerlon+$bmaxlon)/$blonscale;
#my $imgoffsetlat=($ulcornerlat-$ullat)/$blatscale;
my $imgoffsetlat=int(($bmaxlat-$ullat)/$blatscale);
#my $imgoffsetlon=(-$ulcornerlon+$ullon)/$blonscale;
my $imgoffsetlon=int(($ullon-$bminlon)/$blonscale);
if ($debug eq "1") {print "Image Offset Lat:".$imgoffsetlat."<BR>";}
if ($debug eq "1") {print "Image Offset Lon:".$imgoffsetlon."<BR>";}
#if ($debug eq "1") {print "Image Crop Lat:".$cropoffsetlat."<BR>";}
#if ($debug eq "1") {print "Image Crop Lon:".$cropoffsetlon."<BR>";}
#if ($cropoffsetlat>=0) { $cropoffsetlat="+".$cropoffsetlat;}
#if ($cropoffsetlon>=0) { $cropoffsetlon="+".$cropoffsetlon;}
if ($imgoffsetlat>=0) { $imgoffsetlat="+".$imgoffsetlat;}
if ($imgoffsetlon>=0) { $imgoffsetlon="+".$imgoffsetlon;}
# Create the transparent image file.
#my $command="/usr/bin/convert -size ".$wide."x".$high." xc:transparent \"".$bkgtmpimage."\"";

####### Check if we are serving a frame or an image. ##########
if ($getframe eq "1") {
  # We are serving a frame
  if ($debug eq "1") {print "Serving a Frame:  We must Define the Frame in xml";}
  print $query->header("multipart/mixed;boundary=ISRSCOTTWMSVER01010101\r\n");
  print "--ISRSCOTTWMSVER01010101\r\n";
  print $query->header("text/xml");
  print "<BoundingBox SRS=\"$srs\" minx=\"$ullat\" miny=\"$ullon\" maxx=\"$lrlat\" maxy=\"$lrlon\" />\n\r";
  print "--ISRSCOTTWMSVER01010101\r\n";
  my $command="/usr/bin/convert \"".$filename."\" \"".$tmpimage."\"";
  $foo = `$command`;
  $filename=$tmpimage;
  goto END;
}
my $command="/usr/bin/convert -size ".$width."!x".$height."! xc:transparent \"".$bkgtmpimage."\"";
my $junk=`$command`;
if ($debug eq "1") {print "$command<BR>$junk<BR>";}
if ($debug eq "1")
{
my $imageurl=$filename;
$imageurl=~ s/\/maps\/\//;
print "<IMG BORDER=\"10\" SRC=\"$imageurl\"><BR>";
}
# Resize the image file.
$command="/usr/bin/convert -resize ".$imglon."!x".$imglat."! \"".$filename."\" \"".$fgtmpimage."\"";
$junk=`$command`;
if ($debug eq "1") {print "$command<BR>$junk<BR>";}
# Impose the image on the background.
$command="/usr/bin/composite -geometry ".$imgoffsetlon.$imgoffsetlat." \"".$fgtmpimage."\" \"".$bkgtmpimage."\"
\"".$tmpimage."\"";
$junk=`$command`;
if ($debug eq "1") {
  print "$command<BR>$junk<BR>";
  my $imageurl=$fgtmpimage;
  $imageurl=~ s/\/maps\/tmp/\/tmp/;
  print "<IMG BORDER=\"10\" SRC=\"$imageurl\"><BR>";

}
# Crop image to the bounding box size.
#$command="/usr/bin/convert -crop ".$width."x".$height.$cropoffsetlon.$cropoffsetlat." \"".$fgtmpimage."\" \"".$tmpimage."\"";
#$junk=`$command`;
if ($debug eq "1") {print "$command<BR>$junk<BR>";}
```

```
 if ($debug ne "1") {$junk=`rm -rf $fgtmpimage`};
 if ($debug ne "1") {$junk=`rm -rf $bkgtmpimage`};
 $filename=$tmpimage;
 if ($debug eq "1")
   {
   my $imageurl=$filename;
   $imageurl=~ s/\/maps\/tmp/\/tmp/;
   print "<IMG BORDER=\"10\" SRC=\"$imageurl\"><BR>";
   }
}
else
{
 if ($getframe eq "1") {
    # We are serving a frame
    print $query->header("multipart/mixed;boundary=ISRSCOTTWMSVER01010101");
    print "--ISRSCOTTWMSVER01010101\r\n";
    print $query->header("text/xml");
    print "<BoundingBox SRS=\"$srs\" minx=\"$bminlat\" miny=\"$bminlon\" maxx=\"$bmaxlat\" maxy=\"$bmaxlon\" />\n\r";
    print "--ISRSCOTTWMSVER01010101\r\n";
    my $command="/usr/bin/convert \"".$filename."\" \"".$tmpimage."\"";
    $foo = `$command`;
    $filename=$tmpimage;
    goto END;
 }
 my $command="/usr/bin/convert -resize ".$width."x".$height." \"".$filename."\" \"".$tmpimage."\"";
 $foo = `$command`;
 $filename=$tmpimage;
 #print $command."<BR>";
 #print $foo."<BR>";
}
my $command="/usr/bin/convert -transparent white \"".$tmpimage."\" \"".$tmpimage."\"";
#print $command."<BR>";
my $junk=`$command`;
#print $junk."<BR>";
goto END;

GETFRAMES:
if ($debug eq "1") {print $query->header("text/html");}
foreach my $name (@names) {
        if ($name =~ m/^layers$/i) {
                $layer=$query->param($name);
        }
        if ($name =~ m/^format$/i) {
                $format=$query->param($name);
        }
        if ($name =~ m/^width$/i) {
                $width=int($query->param($name));
        }
        if ($name =~ m/^height$/i) {
                $height=int($query->param($name));
        }
        if ($name =~ m/^time$/i) {
                $time=$query->param($name);
        }
        if ($name =~ m/^srs$/i) {
                $srs=$query->param($name);
        }
        if ($name =~ m/^bbox$/i) {
                $bbox=$query->param($name);
```

```
                }
                if ($name =~ m/^styles$/i) {
                        $styles=$query->param($name);
                }
                if ($name =~ m/^transparent$/i) {
                        $transparent=$query->param($name);
                }
        if ($name =~ m/^version$/i) {
              $version=$query->param($name);
        }
}
# Version Number is required.
if ($version eq "0") {
  $filename="/maps/data/errors/nover.png";
  goto END;
}
# Version Number must equal the implementation number.
if ($version ne "1.1.1") {
  $filename="/maps/data/errors/invver.png";
  goto END;
}
# Format is required.
if ($format eq "") {
#  print "No Format";
  $filename="/maps/data/errors/invrequest.png";
  goto END;
}
# Format must be image/png for my server.
if ($format =~ m/^image\/png$/i) {
  #ok
} else {
#  print "Wrong Format";
  $filename="/maps/data/errors/invrequest.png";
  goto END;
}
# Layer is required.
if ($layer eq "") {
#  print "No Layers";
  $filename="/maps/data/errors/invrequest.png";
  goto END;
}
my @jpgfiles;
# Time is required and is a list of time values comma seperated.
if ($time eq "") {
  $time=`ls /maps/data/$layer/*.jpg`;
  @jpgfiles=split(/\n/,$time);
  # Figure out how to parse the directory and return all frames.
} else {
  my @times=split(/,/,$time);
  @jpgfiles=@times;
  for (my $i=0; $i<scalar @times; $i++)
  {
    $jpgfiles[$i]="/maps/data/".$layer."/".$times[$i].".jpg";
  }
}
print $query->header("multipart/mixed;boundary=ISRSCOTTWMSVER01010101");
print "--ISRSCOTTWMSVER01010101\r\n";
foreach $filename (@jpgfiles)
{
  my $d = `date`;
```

```
  $d =~ s/\//_/g;
  $d =~ s/^\s+//;
  $d =~ s/\s+$//;
  $d =~ s/\s/_/g;
  $d =~ s/:/_/g;
  my $tmpimage="/maps/tmp/".$layer.$d.".png";
# Define the bounding box stuff.
    if ($debug eq "1") {print $filename;}
  my $gdalinfo=`gdalinfo \"$filename\"`;
  my @gdallines=split(/\n/,$gdalinfo);
  my $ullon;
  my $ullat;
  my $lrlon;
  my $lrlat;
  foreach my $gdalline (@gdallines){
    if ($gdalline =~ m/Upper Left/){
      $gdalline =~ s/Upper Left//;
      $gdalline =~ s/\(//;
      $gdalline =~ s/\)//;
      ($ullon,$ullat)=split(/,/,$gdalline);
      if ($debug eq "1") {print "Found UpperLeft Lat:".$ullat."<BR>";}
      if ($debug eq "1") {print "Found UpperLeft Lon:".$ullon."<BR>";}
    }
    if ($gdalline =~ m/Lower Right/){
      $gdalline =~ s/Lower Right//;
      $gdalline =~ s/\(//;
      $gdalline =~ s/\)//;
      ($lrlon,$lrlat)=split(/,/,$gdalline);
      if ($debug eq "1") {print "Found LowerRight Lat:".$lrlat."<BR>";}
      if ($debug eq "1") {print "Found LowerRight Lon:".$lrlon."<BR>";}
    }
  }
  if ($debug eq "1") {print "Serving a Frame:  We must Define the Frame in xml";}
  print $query->header("text/xml");
  print "<BoundingBox SRS=\"$srs\" minx=\"$ullat\" miny=\"$ullon\" maxx=\"$lrlat\" maxy=\"$lrlon\" />\n\r";
  print "--ISRSCOTTWMSVER01010101\r\n";
  my $command="/usr/bin/convert \"".$filename."\" \"".$tmpimage."\"";
  $foo = `$command`;
  if ($debug eq "1") {print "$command<BR>\r\n$foo<BR>\r\n";}
  $filename=$tmpimage;
  # write the image file to the output.
  print $query->header("image/png");
  open(IMG, $filename);
  binmode(IMG);
  binmode(STDOUT);
  my @raw_data=<IMG>;
  foreach my $item (@raw_data) { print $item;}
  close(IMG);
  $foo=`rm \"$tmpimage\"`;
  $foo=`rm \"$filename\"`;
  if ($filename ne @jpgfiles[scalar @jpgfiles-1])
  {
    print "\r\n--ISRSCOTTWMSVER01010101\r\n";
  }
}
print "\r\n--ISRSCOTTWMSVER01010101--\r\n";
goto END2;

GETMOVIE:
foreach my $name (@names) {
```

```
                if ($name =~ m/^layers$/i) {
                        $layer=$query->param($name);
                }
                if ($name =~ m/^format$/i) {
                        $format=$query->param($name);
                }
                if ($name =~ m/^width$/i) {
                        $width=int($query->param($name));
                }
                if ($name =~ m/^height$/i) {
                        $height=int($query->param($name));
                }
                if ($name =~ m/^time$/i) {
                        $time=$query->param($name);
                }
                if ($name =~ m/^srs$/i) {
                        $srs=$query->param($name);
                }
                if ($name =~ m/^bbox$/i) {
                        $bbox=$query->param($name);
                }
                if ($name =~ m/^styles$/i) {
                        $styles=$query->param($name);
                }
                if ($name =~ m/^transparent$/i) {
                        $transparent=$query->param($name);
                }
        if ($name =~ m/^version$/i) {
              $version=$query->param($name);
        }
}
print $query->header("multipart/mixed;boundary=ISRSCOTTWMSVER01010101");
print "--ISRSCOTTWMSVER01010101\r\n";
print $query->header("text/xml");
my $xmlname="/maps/data/".$layer."/output.xml";
open(xml, $xmlname);
my @xmldata=<xml>;
foreach my $xmline (@xmldata) { print $xmline;}
close(xml);
print "--ISRSCOTTWMSVER01010101\r\n";
print $query->header("video/avi");
my $moviename="/maps/data/".$layer."/output.avi";
open(IMG, $moviename);
binmode(IMG);
binmode(STDOUT);
my @movie_data=<IMG>;
foreach my $item (@movie_data) { print $item;}
close(IMG);
print "\r\n--ISRSCOTTWMSVER01010101--\r\n";
goto END2;

END:
print $query->header("image/png");
open(IMG, $filename);
binmode(IMG);
binmode(STDOUT);
my @raw_data=<IMG>;
foreach my $item (@raw_data) { print $item;}
close(IMG);
$foo=`rm \"$tmpimage\"`;
```

```
$foo=`rm \"$filename\"`;
if ($getframe eq "1") {
    print "\r\n--ISRSCOTTWMSVER01010101--\r\n";
}
END2: exit;
```

Appendix B Server System Requirements

- Apache Web Server
- Perl 5
- GDAL Toolkit
- Mplayer Toolkit
- Source Code as is will work with LINUX can be modified easily for windows systems. Paths to external tools may need to be changed for everything to work correctly.

Appendix C Sample capabilities XML file.

Should be created as wms.xml on the server.

```
<?xml version='1.0' encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE WMT_MS_Capabilities SYSTEM "http://schemas.opengeospatial.net/wms/1.1.1/WMS_MS_Capabilities.dtd"
 [
 <!ELEMENT VendorSpecificCapabilities EMPTY>
 ]>  <!-- end of DOCTYPE declaration -->

<WMT_MS_Capabilities version="1.1.1">
<Service>
  <Name>Scotts OGC</Name>
  <Title>Scotts OGC</Title>
  <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink" xlink:href="$hostname"/>
</Service>
<Capability>
  <Request>
    <GetCapabilities>
      <Format>application/vnd.ogc.wms_xml</Format>
      <DCPType>
        <HTTP>
          <Get><OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink" xlink:href="$hostname"/></Get>
        </HTTP>
      </DCPType>
    </GetCapabilities>
    <GetMap>
      <Format>image/png</Format>
      <DCPType>
        <HTTP>
          <Get><OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink" xlink:href="$hostname"/></Get>
        </HTTP>
      </DCPType>
    </GetMap>
    <GetFeatureInfo>
      <Format>text/plain</Format>
      <Format>application/vnd.ogc.gml</Format>
      <DCPType>
        <HTTP>
          <Get><OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink" xlink:href="$hostname"/></Get>
        </HTTP>
      </DCPType>
    </GetFeatureInfo>
    <DescribeLayer>
      <Format>text/xml</Format>
      <DCPType>
        <HTTP>
          <Get><OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink" xlink:href="$hostname"/></Get>
        </HTTP>
      </DCPType>
    </DescribeLayer>
    <GetLegendGraphic>
      <Format>image/png</Format>
      <DCPType>
        <HTTP>
          <Get><OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink" xlink:href="$hostname"/></Get>
        </HTTP>
      </DCPType>
    </GetLegendGraphic>
```

```
 </Request>
 <Exception>
  <Format>application/vnd.ogc.se_xml</Format>
  <Format>application/vnd.ogc.se_inimage</Format>
  <Format>application/vnd.ogc.se_blank</Format>
 </Exception>
 <VendorSpecificCapabilities />
 <UserDefinedSymbolization SupportSLD="1" UserLayer="0" UserStyle="1" RemoteWFS="0"/>
 <Layer>
  <Name>Perl WMS</Name>
  <Title>Scott Perl WMS</Title>
  <SRS>EPSG:4326</SRS>
  <LatLonBoundingBox minx="-180" miny="-90" maxx="180" maxy="90" />
  <BoundingBox SRS="EPSG:4326" minx="-180" miny="-90" maxx="180" maxy="90" />
  <Layer>
          <Name>Erin</Name>
          <Title>Hurricane Erin 8-3-1995</Title>
          <LatLonBoundingBox minx="-91" miny="25.24" maxx="-79.36" maxy="32.20" />
          <BoundingBox SRS="EPSG:4326" minx="-91" miny="25.24" maxx="-79.36" maxy="32.20"/>
          <Dimension name="time" units="ISO8601"/>
          <Extent name="time" default="1995-08-03 1400Z" nearestValue="0">1995-08-03 1400Z,1995-08-03 1500Z,1995-08-03
1600Z,1995-08-03 1700Z</Extent>
  </Layer>
  <Layer>
          <Name>ErinTrack</Name>
          <Title>Huruicane Erin Path</Title>
          <LatLonBoundingBox minx="-92.70" miny="19.60" maxx="-70.80" maxy="35.60" />
          <BoundingBox SRS="EPSG:4326"  minx="-92.70" miny="19.60" maxx="-70.80" maxy="35.60" />
          <Dimension name="time" units="ISO8601"/>
          <Extent name="time" default="1995-07-31T03Z" nearestValue="0">1995-07-31T03Z/1995-08-04T09Z/PT3H</Extent>
  </Layer>
  <Layer>
    <Name>moon</Name>
    <Title>Earth's Moon Surface</Title>
          <LatLonBoundingBox minx="-180" miny="-90" maxx="180" maxy="90" />
          <BoundingBox SRS="EPSG:4326"  minx="-89.60" miny="22.50" maxx="-73.70" maxy="32.70" />
  </Layer>
  <Layer>
    <Name>Mars</Name>
    <Title>Mar's Surface</Title>
          <LatLonBoundingBox minx="-180" miny="-90" maxx="180" maxy="90" />
          <BoundingBox SRS="EPSG:4326"  minx="-89.60" miny="22.50" maxx="-73.70" maxy="32.70" />
  </Layer>
  <Layer>
    <Name>Sun</Name>
    <Title>Sun's Magnetic Field</Title>
          <LatLonBoundingBox minx="-180" miny="-90" maxx="180" maxy="90" />
          <BoundingBox SRS="EPSG:4326"  minx="-89.60" miny="22.50" maxx="-73.70" maxy="32.70" />
          <Dimension name="time" units="ISO8601"/>
          <Extent name="time" default="1835"
nearestValue="0">0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,
41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78</Extent>
   </Layer>
   <Layer>
          <Name>Charley</Name>
          <Title>Huruicane Charley Animation</Title>
          <LatLonBoundingBox minx="-90.5253" miny="12.6028" maxx="-68.8210" maxy="34.2722" />
          <BoundingBox SRS="EPSG:4326"  minx="-90.5253" miny="12.6028" maxx="-68.8210" maxy="34.2722" />
          <Dimension name="time" units="ISO8601"/>
```

```
        <Extent name="time" default="2004-08-11T18:15Z" nearestValue="0">2004-08-11T18:15Z,2004-08-12T15:55Z,2004-
08-13T16:35Z</Extent>
    </Layer>
    <Layer>
      <Name>Mt_St_Helens</Name>
      <Title>Mount St. Helens Before, During, and After (1024x1024 Animation)</Title>
      <Abstract>Mount St. Helens erupted on May 18, 1980, devastating more than 150 square miles of forest in southwestern
Washington state.  This animation shows Landsat images of the Mount St. Helens area in 1973, 1983, and 2000, illustrating the
destruction and regrowth of the forest.  The 1983 image clearly shows the new crater on the northern slope where the eruption
occurred, the rivers and lakes covered with ash, and the regions of deforestation.  The 2000 image, taken twenty years after the
eruption, still shows the changed crater, but much of the devastated area is covered by new vegetation growth.</Abstract>
      <LatLonBoundingBox minx="-122.36011" miny="46.122115" maxx="-121.97916" maxy="46.384547" />
      <BoundingBox SRS="EPSG:4326"  minx="-122.36011" miny="46.122115" maxx="-121.97916" maxy="46.384547" />
      <Dimension name="time" units="ISO8601"/>
        <Extent name="time" default="09-25-2000">09-15-1973,05-22-1983,09-25-2000</Extent>
    </Layer>
    <Layer>
      <Name>Katrina</Name>
      <Title> Huricane Katrina Landfall Event </Title>
      <LatLonBoundingBox minx="-101" miny="17.5" maxx="-75" maxy="32.5" />
      <BoundingBox SRS="EPSG:4326" minx="-101" miny="17.5" maxx="-75" maxy="32.5" />
      <Dimension name="time" units="ISO8601"/>
      <Extent name="time" default="0001"
nearestValue="0">0001,0002,0003,0004,0005,0006,0007,0008,0009,0010,0011,0012,0013,0014,0015,0016,0017,0018,0019,0020,002
1,0022,0023,0024,0025,0026,0027,0028,0029,0030,0031,0032,0033,0034,0035,0036,0037,0038,0039,0040,0041,0042,0043,0044,0
045,0046,0047,0048,0049,0050,0051,0052,0053,0054,0055,0056,0057,0058,0059,0060,0061,0062,0063,0064,0065,066,0067,0068,
0069,0070,0071,072,0073,0074,0075,0076,0077,0078,0079,0080,0081,0082,0083,0084,0085,0086,0087,0088,0089,0090,0091,0092
,0-
093,0094,0095,0096,0097,0098,0099,0100,0101,0102,0103,0104,0105,0106,0107,0108,0109,0110,0111,0112,0113,0114,0115,0116,011
7,0118,0119,0120,0121,0122,0123,0124,0125,0126,0127,0128,0129,0130,0131,0132,0133,0134,0135,0136,0137,0138,0139,0140,0141,0
142,0143,0144,0145,0146,0147,0148,0149,0150,0151,0152,0153,0154,0155,0156,0157,0158,0159,0160,0161,0162,0163,0164,0165,016
6,0167,0168,0169,0170,0171,0172,0173,0174,0175,0176,0177,0178,0179,0180,0181,0182,0183,0184,0185,0186,0187,0188,0189,0190,01
91,0192,0193,0194,0195,0196,0197,0198,0199,0200,0201,0202,0203,0204,0205,0206,0207,0208,0209,0210,0211,0212,0213,0214,02
15,0216,0217,0218,0219,0220,0221,0222,0223,0224,0225,0226,0227,0228,0229,0230,0231,0232,0233,0234,0235,0236,0237,0238,0
239,0240,0241,0242,0243,0244</Extent>
    </Layer>
  </Layer>
</Capability>
</WMT_MS_Capabilities>
```

Appendix D Client Modifications.

Download Thread difference file:

```
22a23,36
>
>                       public WMSDownload(WMSDownload wdl)
>                       {
>                               this._date = wdl.Date;
>                               this._east = wdl.East;
>                               this._name = wdl.Name;
>                               this._north = wdl.North;
>                               this._savedFilePath = wdl.SavedFilePath;
>                               this._south = wdl.South;
>                               this._title = wdl.Title;
>                               this._url = wdl.Url;
>                               this._version = wdl.Version;
>                               this._west = wdl.West;
>                       }
```

WMSBrowser difference file:

```
15a16
> using System.Text.RegularExpressions;
27d27
<                       private System.Windows.Forms.Button getFramesBtn;
30c30
<                       private System.Windows.Forms.Button button3;
---
>                       private System.Windows.Forms.Label labelEast;
39c39
<                       private System.Windows.Forms.Label labelEast;
---
>                       private System.Windows.Forms.GroupBox groupBoxLayerOptions;
49c49
<                       private System.Windows.Forms.NumericUpDown numericUpDownNorth;
---
>                       private System.Windows.Forms.Button GetMovieBtn;
52a53
>                       private System.Windows.Forms.NumericUpDown numericUpDownNorth;
59d59
<                       private System.Windows.Forms.GroupBox groupBoxLayerOptions;
60a61
>                       private System.Windows.Forms.Button getMultipleFramesBtn;
69a71
>                       private System.Windows.Forms.Button GetFramesBtn;
101a104,105
>                       private Thread framesThread;
>                       private Thread movieThread;
107c111,112
<
---
>                       private System.Collections.Queue frameQueue = new Queue();
>                       private System.Collections.Queue movieQueue = new Queue();
122d126
<
125c129
<                               //this.animationTimer.Start();
---
```

```
>                               this.animationTimer.Start();
128a133,148
>
>                       void StartMovieThread()
>                       {
>                               if (movieThread!=null && movieThread.IsAlive)
>                                       return;
>                               //MessageBox.Show("The movie download thread has started.", "Frames Thread Starting",
MessageBoxButtons.OK, MessageBoxIcon.None, MessageBoxDefaultButton.Button1);
>                               movieThread = new Thread( new ThreadStart(MovieDownloader));
>                               movieThread.Name = "WMSBrowser.MovieDownloader";
>                               movieThread.IsBackground = true;
>                               movieThread.Start();
>                       }
>
>                       void StopMovieThread()
>                       {
>                               if (movieThread==null || !movieThread.IsAlive)
>                                       return;
129a150,170
>                               movieThread.Abort();
>                       }
>
>                       void StartFramesThread()
>                       {
>                               if (framesThread!=null && framesThread.IsAlive)
>                                       return;
>                               //MessageBox.Show("The frames download thread has started.", "Frames Thread Starting",
MessageBoxButtons.OK, MessageBoxIcon.None, MessageBoxDefaultButton.Button1);
>                               framesThread = new Thread( new ThreadStart(FrameDownloader));
>                               framesThread.Name = "WMSBrowser.FrameDownloader";
>                               framesThread.IsBackground = true;
>                               framesThread.Start();
>                       }
>
>                       void StopFramesThread()
>                       {
>                               if (framesThread==null || !framesThread.IsAlive)
>                                       return;
>
>                               framesThread.Abort();
>                       }
146a188,335
>
>                       private void MovieDownloader()
>                       {
>                               WMSDownload wmsDownload = null;
>                               while(true)
>                               {
>                                       try
>                                       {
>                                               lock(this.movieQueue)
>                                               {
>                                                       if(this.movieQueue.Count <= 0)
>                                                               // Queue empty
>                                                               return;
>                                                       wmsDownload = (WMSDownload)this.movieQueue.Dequeue();
>                                               }
>
```

```
>                                                       UpdateStatusBar( "Downloading: " +
Path.GetFileName(wmsDownload.SavedFilePath) );
>                                                       try{
>                                                       using( WebDownload dl = new WebDownload(wmsDownload.Url) )
>                                                       {
>                                                               //MessageBox.Show(wmsDownload.Url, "Downloading MHT
File", MessageBoxButtons.OK, MessageBoxIcon.None, MessageBoxDefaultButton.Button1);
>
        Directory.CreateDirectory(Path.GetDirectoryName(wmsDownload.SavedFilePath));
>                                                       string mhtfile =
Path.Combine(Path.GetDirectoryName(wmsDownload.SavedFilePath),"movie.mht");
>                                                       //string mhtfile = wdl.SavedFilePath + ".mht";
>                                                       dl.ProgressCallback += new
DownloadProgressHandler(this.updateCurrentProgressBar);
>                                                       dl.DownloadFile(mhtfile);
>
>                                                       //dl.DownloadMemory(DownloadType.Wms);
>                                                       if(this.animationState == AnimationState.Cancel ||
this.animationState == AnimationState.Stop)
>                                                               return;
>
>                                                       dl.Verify();
>                                                       if (dl.ContentType.Substring(0,9) == "multipart" )
>                                                       {
>                                                               //MessageBox.Show(dl.ContentType.Substring(25),
"Seperation String", MessageBoxButtons.OK, MessageBoxIcon.None, MessageBoxDefaultButton.Button1);
>                                                               ProcessDownloadedMovie(wmsDownload, dl);
>                                                       }
>                                                       else
>                                                       {
>                                                               MessageBox.Show("Server did not return correct
MIME type.", "Error Downloading MHT File", MessageBoxButtons.OK, MessageBoxIcon.None,
MessageBoxDefaultButton.Button1);
>                                                       }
>                                               }
>                                               }
>                                               catch (Exception E)
>                                               {
>                                                       MessageBox.Show(E.Message+E.StackTrace, "Error
Downloading MHT File", MessageBoxButtons.OK, MessageBoxIcon.None, MessageBoxDefaultButton.Button1);
>
>                                                       }
>                                       }
>                                       catch(WebException caught)
>                                       {
>                                               this.worldWindow.Caption = caught.Message;
>                                       }
>                                       finally
>                                       {
>                                               updateCurrentProgressBar(0,1);
>                                       }
>                               }
>                       }
>
>                       private void FrameDownloader()
>                       {
>                               WMSDownload wmsDownload = null;
>                               while(true)
>                               {
>                                       try
```

```
>                                       {
>                                               lock(this.frameQueue)
>                                               {
>                                                       if(this.frameQueue.Count <= 0)
>                                                               // Queue empty
>                                                               return;
>                                                       wmsDownload = (WMSDownload)this.frameQueue.Dequeue();
>                                               }
>
>
>                                               if(this.checkBoxAnimationCache.Checked &&
>                                                       File.Exists(wmsDownload.SavedFilePath + ".wwi") &&
>                                                       File.Exists(wmsDownload.SavedFilePath + ".dds"))
>                                               {
>                                                       try
>                                                       {
>                                                               ImageLayerInfo imageLayerInfo =
ImageLayerInfo.FromFile(wmsDownload.SavedFilePath + ".wwi");
>                                                               const float epsilon = Single.Epsilon*100;
>                                                               if(Math.Abs(imageLayerInfo.North -
(float)wmsDownload.North)<epsilon &&
>                                                                       Math.Abs(imageLayerInfo.South -
(float)wmsDownload.South)<epsilon &&
>                                                                       Math.Abs(imageLayerInfo.West -
(float)wmsDownload.West)<epsilon &&
>                                                                       Math.Abs(imageLayerInfo.East -
(float)wmsDownload.East)<epsilon)
>                                                               {
>                                                                       lock(this.animationFrames.SyncRoot)
>                                                                       {
>
        animationFrames.Add(imageLayerInfo);
>                                                                               animationLayers.Add(null);
>                                                                               // Process next queue item
>                                                                               continue;
>                                                                       }
>                                                               }
>                                                       }
>                                                               // Cached file not readable for some reason - reload
>                                                       catch(IOException) {}
>                                                       catch(FormatException) {}
>                                               }
>
>                                               UpdateStatusBar( "Downloading: " +
Path.GetFileName(wmsDownload.SavedFilePath) );
>                                               try{
>                                               using( WebDownload dl = new WebDownload(wmsDownload.Url) )
>                                               {
>                                                       //MessageBox.Show(wmsDownload.Url, "Downloading MHT
File", MessageBoxButtons.OK, MessageBoxIcon.None, MessageBoxDefaultButton.Button1);
>
        Directory.CreateDirectory(Path.GetDirectoryName(wmsDownload.SavedFilePath));
>                                                       string mhtfile =
Path.Combine(Path.GetDirectoryName(wmsDownload.SavedFilePath),"frames.mht");
>                                                       //string mhtfile = wdl.SavedFilePath + ".mht";
>                                                       dl.ProgressCallback += new
DownloadProgressHandler(this.updateCurrentProgressBar);
>                                                       dl.DownloadFile(mhtfile);
>
>                                                       //dl.DownloadMemory(DownloadType.Wms);
```

```
>                                                       if(this.animationState == AnimationState.Cancel ||
this.animationState == AnimationState.Stop)
>                                                               return;
>
>                                                       dl.Verify();
>                                                       if (dl.ContentType.Substring(0,9) == "multipart" )
>                                                       {
>                                                               //MessageBox.Show(dl.ContentType.Substring(25),
"Seperation String", MessageBoxButtons.OK, MessageBoxIcon.None, MessageBoxDefaultButton.Button1);
>                                                               ProcessDownloadedFrames(wmsDownload, dl);
>                                                       }
>                                                       else
>                                                       {
>
>                                                               ProcessDownloadedImage(wmsDownload,
dl.ContentStream);
>                                                       }
>                                               }
>                                               }
>                                               catch (Exception E)
>                                               {
>
>                                                       MessageBox.Show(E.Message+E.StackTrace, "Error
Downloading MHT File", MessageBoxButtons.OK, MessageBoxIcon.None, MessageBoxDefaultButton.Button1);
>                                                       }
>                                       }
>                                       catch(WebException caught)
>                                       {
>                                               this.worldWindow.Caption = caught.Message;
>                                       }
>                                       finally
>                                       {
>                                               updateCurrentProgressBar(0,1);
>                                       }
>                               }
>                       }
>
187a377
>                               this.GetFramesBtn = new System.Windows.Forms.Button();
196a387
>                               this.getMultipleFramesBtn = new System.Windows.Forms.Button();
198d388
<                               this.groupBoxLayerOptions = new System.Windows.Forms.GroupBox();
204a395
>                               this.numericUpDownNorth = new System.Windows.Forms.NumericUpDown();
208c399
<                               this.numericUpDownNorth = new System.Windows.Forms.NumericUpDown();
---
>                               this.GetMovieBtn = new System.Windows.Forms.Button();
218c409
<                               this.labelEast = new System.Windows.Forms.Label();
---
>                               this.groupBoxLayerOptions = new System.Windows.Forms.GroupBox();
227c418
<                               this.button3 = new System.Windows.Forms.Button();
---
>                               this.labelEast = new System.Windows.Forms.Label();
230d420
<                               this.getFramesBtn = new System.Windows.Forms.Button();
239d428
```

```
<                       this.groupBoxLayerOptions.SuspendLayout();
243d431
<                       this.tabControl1.SuspendLayout();
244a433
>                       this.tabControl1.SuspendLayout();
247a437
>                       this.groupBoxLayerOptions.SuspendLayout();
340a531,539
>                       // GetFramesBtn
>                       //
>                       this.GetFramesBtn.Location = new System.Drawing.Point(144, 120);
>                       this.GetFramesBtn.Name = "GetFramesBtn";
>                       this.GetFramesBtn.Size = new System.Drawing.Size(48, 32);
>                       this.GetFramesBtn.TabIndex = 8;
>                       this.GetFramesBtn.Text = "GetFrame";
>                       this.GetFramesBtn.Click += new System.EventHandler(this.GetFramesBtnClick);
>                       //
372a572,573
>                       this.tabPageAnimation.Controls.Add(this.GetMovieBtn);
>                       this.tabPageAnimation.Controls.Add(this.getMultipleFramesBtn);
382,383c583
<                       this.tabPageAnimation.Controls.Add(this.getFramesBtn);
<                       this.tabPageAnimation.Controls.Add(this.button3);
---
>                       this.tabPageAnimation.Controls.Add(this.GetFramesBtn);
468a669,676
>                       // getMultipleFramesBtn
>                       //
>                       this.getMultipleFramesBtn.Location = new System.Drawing.Point(128, 96);
>                       this.getMultipleFramesBtn.Name = "getMultipleFramesBtn";
>                       this.getMultipleFramesBtn.TabIndex = 0;
>                       this.getMultipleFramesBtn.Text = "GetFrames";
>                       this.getMultipleFramesBtn.Click += new System.EventHandler(this.GetMultipleFramesBtnClick);
>                       //
479,495d686
<                       // groupBoxLayerOptions
<                       //
<                       this.groupBoxLayerOptions.Controls.Add(this.labelPercent);
<                       this.groupBoxLayerOptions.Controls.Add(this.labelKM);
<                       this.groupBoxLayerOptions.Controls.Add(this.labelLegend);
<                       this.groupBoxLayerOptions.Controls.Add(this.numericUpDownHeight);
<                       this.groupBoxLayerOptions.Controls.Add(this.numericUpDownTransparency);
<                       this.groupBoxLayerOptions.Controls.Add(this.labelHeight);
<                       this.groupBoxLayerOptions.Controls.Add(this.labelTransparency);
<                       this.groupBoxLayerOptions.Controls.Add(this.buttonLegend);
<                       this.groupBoxLayerOptions.Location = new System.Drawing.Point(272, 120);
<                       this.groupBoxLayerOptions.Name = "groupBoxLayerOptions";
<                       this.groupBoxLayerOptions.Size = new System.Drawing.Size(216, 96);
<                       this.groupBoxLayerOptions.TabIndex = 2;
<                       this.groupBoxLayerOptions.TabStop = false;
<                       this.groupBoxLayerOptions.Text = "Options";
<                       //
558a750,773
>                       // numericUpDownNorth
>                       //
>                       this.numericUpDownNorth.DecimalPlaces = 2;
>                       this.numericUpDownNorth.Location = new System.Drawing.Point(48, 20);
>                       this.numericUpDownNorth.Maximum = new System.Decimal(new int[] {
>                                                         90,
>                                                         0,
```

```
>                                                   0,
>                                                   0});
>                           this.numericUpDownNorth.Minimum = new System.Decimal(new int[] {
>                                                   90,
>                                                   0,
>                                                   0,
>                                                   -2147483648});
>                           this.numericUpDownNorth.Name = "numericUpDownNorth";
>                           this.numericUpDownNorth.Size = new System.Drawing.Size(56, 20);
>                           this.numericUpDownNorth.TabIndex = 1;
>                           this.numericUpDownNorth.TextAlign = System.Windows.Forms.HorizontalAlignment.Right;
>                           this.numericUpDownNorth.Value = new System.Decimal(new int[] {
>                                                   90,
>                                                   0,
>                                                   0,
>                                                   0});
>                           //
590c805
<                           // numericUpDownNorth
---
>                           // GetMovieBtn
592,612c807,812
<                           this.numericUpDownNorth.DecimalPlaces = 2;
<                           this.numericUpDownNorth.Location = new System.Drawing.Point(48, 20);
<                           this.numericUpDownNorth.Maximum = new System.Decimal(new int[] {
<                                                   90,
<                                                   0,
<                                                   0,
<                                                   0});
<                           this.numericUpDownNorth.Minimum = new System.Decimal(new int[] {
<                                                   90,
<                                                   0,
<                                                   0,
<                                                   -2147483648});
<                           this.numericUpDownNorth.Name = "numericUpDownNorth";
<                           this.numericUpDownNorth.Size = new System.Drawing.Size(56, 20);
<                           this.numericUpDownNorth.TabIndex = 1;
<                           this.numericUpDownNorth.TextAlign = System.Windows.Forms.HorizontalAlignment.Right;
<                           this.numericUpDownNorth.Value = new System.Decimal(new int[] {
<                                                   90,
<                                                   0,
<                                                   0,
<                                                   0});
---
>                           this.GetMovieBtn.Location = new System.Drawing.Point(56, 96);
>                           this.GetMovieBtn.Name = "GetMovieBtn";
>                           this.GetMovieBtn.Size = new System.Drawing.Size(72, 24);
>                           this.GetMovieBtn.TabIndex = 9;
>                           this.GetMovieBtn.Text = "Get Movie";
>                           this.GetMovieBtn.Click += new System.EventHandler(this.GetMovieBtnClick);
725c925
<                           // labelEast
---
>                           // groupBoxLayerOptions
727,732c927,940
<                           this.labelEast.Location = new System.Drawing.Point(106, 41);
<                           this.labelEast.Name = "labelEast";
<                           this.labelEast.Size = new System.Drawing.Size(40, 23);
<                           this.labelEast.TabIndex = 6;
<                           this.labelEast.Text = "&East:";
```

```
<                       this.labelEast.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
---
>                       this.groupBoxLayerOptions.Controls.Add(this.labelPercent);
>                       this.groupBoxLayerOptions.Controls.Add(this.labelKM);
>                       this.groupBoxLayerOptions.Controls.Add(this.labelLegend);
>                       this.groupBoxLayerOptions.Controls.Add(this.numericUpDownHeight);
>                       this.groupBoxLayerOptions.Controls.Add(this.numericUpDownTransparency);
>                       this.groupBoxLayerOptions.Controls.Add(this.labelHeight);
>                       this.groupBoxLayerOptions.Controls.Add(this.labelTransparency);
>                       this.groupBoxLayerOptions.Controls.Add(this.buttonLegend);
>                       this.groupBoxLayerOptions.Location = new System.Drawing.Point(272, 120);
>                       this.groupBoxLayerOptions.Name = "groupBoxLayerOptions";
>                       this.groupBoxLayerOptions.Size = new System.Drawing.Size(216, 96);
>                       this.groupBoxLayerOptions.TabIndex = 2;
>                       this.groupBoxLayerOptions.TabStop = false;
>                       this.groupBoxLayerOptions.Text = "Options";
818c1026
<                       // button3
---
>                       // labelEast
820,824c1028,1033
<                       this.button3.Location = new System.Drawing.Point(144, 104);
<                       this.button3.Name = "button3";
<                       this.button3.Size = new System.Drawing.Size(48, 32);
<                       this.button3.TabIndex = 8;
<                       this.button3.Text = "button1";
---
>                       this.labelEast.Location = new System.Drawing.Point(106, 41);
>                       this.labelEast.Name = "labelEast";
>                       this.labelEast.Size = new System.Drawing.Size(40, 23);
>                       this.labelEast.TabIndex = 6;
>                       this.labelEast.Text = "&East:";
>                       this.labelEast.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
843,851d1051
<                       // getFramesBtn
<                       //
<                       this.getFramesBtn.Location = new System.Drawing.Point(144, 104);
<                       this.getFramesBtn.Name = "getFramesBtn";
<                       this.getFramesBtn.Size = new System.Drawing.Size(48, 48);
<                       this.getFramesBtn.TabIndex = 8;
<                       this.getFramesBtn.Text = "Frames";
<                       this.getFramesBtn.Click += new System.EventHandler(this.GetFramesBtnClick);
<                       //
868d1067
<
888d1086
<                       this.groupBoxLayerOptions.ResumeLayout(false);
892d1089
<                       this.tabControl1.ResumeLayout(false);
893a1091
>                       this.tabControl1.ResumeLayout(false);
896a1095
>                       this.groupBoxLayerOptions.ResumeLayout(false);
1173a1373,1377
>                       lock(this.frameQueue.SyncRoot)
>                               this.frameQueue.Clear();
>
>                       StopFramesThread();
>
1557a1762
```

```
>
1563c1768
<                       private string ReadBinLine(BinaryReader br)
---
>                       private string ReadimageLine(BinaryReader br)
1566a1772,1803
>                               int count=0;
>                               while ((c != '\n')&(c != '\r'))
>                               {
>                                       this.UpdateStatusBar( line );
>                                       //MessageBox.Show(line, "Process Header Line", MessageBoxButtons.OK,
MessageBoxIcon.None, MessageBoxDefaultButton.Button1);
>                                       line=line + c.ToString();
>                                       c = br.ReadChar();
>                                       count++;
>                                       if (count > 256)
>                                       {
>                                               //MessageBox.Show("The Line Won't end!!!", "I am stuck here",
MessageBoxButtons.OK, MessageBoxIcon.None, MessageBoxDefaultButton.Button1);
>                                               line=line + c.ToString();
>                                               return line;
>                                       }
>
>                               }
>                               line= line+c.ToString();
>                               return line;
>                       }
>
>                       /// <summary>
>                       /// Implements read line capability for binary files.
>                       /// </summary>
>                       /// <param name="br"></param>
>                       /// <returns></returns>
>                       private string ReadBinLine(BinaryReader br)
>                       {
>                               string line="";
>                               try
>                               {
>                               char c = br.ReadChar();
>
1570c1807
<                                       {
---
>                                       {
1575a1813,1820
>                               } catch (EndOfStreamException E)//Special case for end of stream
>                               {
>                                       //MessageBox.Show(E.Message, "Read Line Error", MessageBoxButtons.OK,
MessageBoxIcon.None, MessageBoxDefaultButton.Button1);
>                                       return null;
>                               } catch { //This is here for lines that seem to never end.
>                                       return line;
>                               }
>
1576a1822
>
1577a1824
>
1583,1590d1829
```

```
<                                       //MessageBox.Show("Processing Frame Content ", "Processss Frame", MessageBoxButtons.OK,
MessageBoxIcon.None, MessageBoxDefaultButton.Button1);
<                                       //dl.DownloadFile("c:/tempfile.mhtml");
<                                       //MessageBox.Show("Processing Frame Content "+dl.ContentType, "Processss Frame",
MessageBoxButtons.OK, MessageBoxIcon.None, MessageBoxDefaultButton.Button1);
<                                       //Stream fs = new FileStream("c:/tempfile.mhtml",FileMode.Open);
<                                       //decimal North = wdl.North;
<                                       //decimal West = wdl.West;
<                                       //decimal South = wdl.South;
<                                       //decimal East = wdl.East;
1668c1907,2304
<                                       datastream.Close();
---
>                                       datastream.Close();
>                               }
>
>                               /// <summary>
>                               /// Convert requested frame and prepare downloaded data for display.
>                               /// </summary>
>                               private void ProcessDownloadedFrames(WMSDownload wdl, WebDownload dl)
>                               {
>                                       //MessageBox.Show(wdl.SavedFilePath, "Downloading MHT File", MessageBoxButtons.OK,
MessageBoxIcon.None, MessageBoxDefaultButton.Button1);
>                                       //Directory.CreateDirectory(Path.GetDirectoryName(wdl.SavedFilePath));
>                                       string mhtfile = Path.Combine(Path.GetDirectoryName(wdl.SavedFilePath),"frames.mht");
>                                       //string mhtfile = wdl.SavedFilePath + ".mht";
>                                       string seperator=dl.ContentType.Substring(25);
>                                       //dl.DownloadFile(mhtfile);
>                                       string[] filename = wdl.Date.Split(',');
>                                       BinaryReader br = new BinaryReader(new FileStream(mhtfile,FileMode.Open));
>                                       string line = ReadBinLine(br);
>                                       ArrayList Lines = new ArrayList();
>                                       int count=0;
>                                       int imgstart=0;
>                                       int imgend=0;
>                                       while (line != null)
>                                       {
>                                               string filetype="";
>                                               while (!(Regex.Match(line,seperator).Success))
>                                               {
>                                                       //MessageBox.Show(line, "Processing MHT File", MessageBoxButtons.OK,
MessageBoxIcon.None, MessageBoxDefaultButton.Button1);
>                                                       if (Regex.Match(line,"text/").Success)
>                                                       {
>                                                               filetype="xml";
>                                                       }
>                                                       else
>                                                       {
>                                                               if (Regex.Match(line,"image/").Success)
>                                                               {
>                                                                       filetype="PNG";
>                                                                       imgstart=(int)(br.BaseStream.Position);
>                                                               }
>                                                               else
>                                                               {
>                                                                       Lines.Add(line);
>                                                               }
>                                                       }
>                                                       line = ReadBinLine(br);
>                                               }
```

```
>                                       //MessageBox.Show("Finished Reading a File Segment of "+Lines.Count.ToString()+"
lines.", "Processing MHT File", MessageBoxButtons.OK, MessageBoxIcon.None, MessageBoxDefaultButton.Button1);
>
>                                       //string[] lines = (string[])(Lines.ToArray(typeof(string)));
>                                       if (filetype.Equals("xml"))
>                                       {
>                                               for (int i=0; i<Lines.Count; i++)
>                                               {
>                                                       string lin=(string)(Lines[i]);
>                                                       if (Regex.Match(lin,"BoundingBox").Success)
>                                                       {
>                                                               //MessageBox.Show("Begin Processing xml" ,
"Processing xml File", MessageBoxButtons.OK, MessageBoxIcon.None, MessageBoxDefaultButton.Button1);
>                                                               string[] temp = lin.Split("");
>                                                               wdl.North = decimal.Parse(temp[3]);
>                                                               wdl.West = decimal.Parse(temp[5]);
>                                                               wdl.South = decimal.Parse(temp[7]);
>                                                               wdl.East = decimal.Parse(temp[9]);
>                                                       }
>                                               }
>                                       }
>                                       if (filetype.Equals("PNG"))
>                                       {
>                                               //MessageBox.Show(wdl.SavedFilePath, "Processing PNG File",
MessageBoxButtons.OK, MessageBoxIcon.None, MessageBoxDefaultButton.Button1);
>                                               imgend=(int)(br.BaseStream.Position);
>                                               int imgsize = imgend-imgstart;
>                                               byte[] img = new byte[imgsize];
>                                               br.BaseStream.Seek((long)(imgstart+2),SeekOrigin.Begin);
>                                               br.Read(img,0,img.Length);
>                                               string path = Path.GetDirectoryName(wdl.SavedFilePath);
>                                               if(filename[count]!=null && filename[count].Length>0)
>                                               {
>                                                       wdl.SavedFilePath = Path.Combine( path,
filename[count].Replace(":",""));
>                                                       wdl.Title = "\n" + filename[count];
>                                               }
>                                               else
>                                                       wdl.SavedFilePath = Path.Combine( path, "Default" );
>                                               count++;
>                                               FileStream fs = new
FileStream(wdl.SavedFilePath+".png",FileMode.OpenOrCreate);
>                                               BinaryWriter bw = new BinaryWriter(fs);
>                                               bw.Write(img,0,img.Length);
>                                               bw.Close();
>                                               //MessageBox.Show(wdl.SavedFilePath, "Done Processing PNG File",
MessageBoxButtons.OK, MessageBoxIcon.None, MessageBoxDefaultButton.Button1);
>
>                                               string ddsimage = wdl.SavedFilePath + ".png";
>                                               Stream datastream = new FileStream(ddsimage,FileMode.Open);
>                                               try
>                                               {
>                                                       string ddsFile = wdl.SavedFilePath + ".dds";
>                                                       if( wdl.Date.Length>0)
>                                                               UpdateStatusBar( "Converting " + wdl.Date );
>                                                       else
>                                                               UpdateStatusBar( "Converting... " );
>
        Directory.CreateDirectory(Path.GetDirectoryName(wdl.SavedFilePath));
>                                                       if (datastream.Length==0)
```

```
>                                                                       throw new WebException("Server returned no data.");
>                                                               ImageHelper.ConvertToDxt3(datastream, ddsFile,
worldWindow.DrawArgs.device );
>                                                               if(this.animationState == AnimationState.Cancel || animationState
== AnimationState.Stop)
>                                                                       return;
>                                                               ImageLayerInfo imageLayerInfo = new ImageLayerInfo(wdl);
>                                                               imageLayerInfo.Save(wdl.SavedFilePath + ".wwi");
>
>                                                               lock(this.animationFrames.SyncRoot)
>                                                                       animationFrames.Add(imageLayerInfo);
>                                                               animationLayers.Add(null);
>
>                                                               if(this.animationState == AnimationState.Cancel ||
this.animationState == AnimationState.Stop)
>                                                                       return;
>
>                                                               UpdateStatusBar( "" );
>                                                       }
>                                                       catch(Exception caught)
>                                                       {
>                                                               this.worldWindow.Caption = caught.Message;
>                                                               Utility.Log.Write( caught );
>                                                               MessageBox.Show(caught.ToString(), "Frame Image Extraction
Error", MessageBoxButtons.OK, MessageBoxIcon.None, MessageBoxDefaultButton.Button1);
>                                                       }
>                                                       datastream.Close();
>
>                                               }
>                                               Lines.Clear();
>                                               filetype="";
>                                               line=ReadBinLine(br);
>                                       }
>                                       //MessageBox.Show("Completed Layer Extraction", "Processing MHT File",
MessageBoxButtons.OK, MessageBoxIcon.None, MessageBoxDefaultButton.Button1);
>
>                                       br.Close();
>                       }
>
>                       /// <summary>
>                       /// Convert requested frame and prepare downloaded data for display.
>                       /// </summary>
>                       private void ProcessDownloadedMovie(WMSDownload wdl, WebDownload dl)
>                       {
>                                       //MessageBox.Show(wdl.SavedFilePath, "Downloading MHT File", MessageBoxButtons.OK,
MessageBoxIcon.None, MessageBoxDefaultButton.Button1);
>                                       //Directory.CreateDirectory(Path.GetDirectoryName(wdl.SavedFilePath));
>                                       string mhtfile = Path.Combine(Path.GetDirectoryName(wdl.SavedFilePath),"movie.mht");
>                                       //string mhtfile = wdl.SavedFilePath + ".mht";
>                                       string seperator=dl.ContentType.Substring(25);
>                                       //dl.DownloadFile(mhtfile);
>                                       string[] filename = wdl.Date.Split(',');
>                                       BinaryReader br = new BinaryReader(new FileStream(mhtfile,FileMode.Open));
>                                       string line = ReadBinLine(br);
>                                       ArrayList Lines = new ArrayList();
>                                       int imgstart=0;
>                                       int imgend=0;
>                                       ArrayList FramesData = new ArrayList();
>                                       while (line != null)
>                                       {
```

```
>                                       string filetype="";
>                                       while (!(Regex.Match(line,seperator).Success))
>                                       {
>                                               //MessageBox.Show(line, "Processing MHT File", MessageBoxButtons.OK,
MessageBoxIcon.None, MessageBoxDefaultButton.Button1);
>                                               if (Regex.Match(line,"text/").Success)
>                                               {
>                                                       filetype="xml";
>                                               }
>                                               else
>                                               {
>                                                       if (Regex.Match(line,"video/").Success)
>                                                       {
>                                                               filetype="AVI";
>                                                               imgstart=(int)(br.BaseStream.Position);
>                                                       }
>                                                       else
>                                                       {
>                                                               Lines.Add(line);
>                                                       }
>                                               }
>                                               line = ReadBinLine(br);
>                                       }
>                                       //MessageBox.Show("Finished Reading a File Segment of "+Lines.Count.ToString()+"
lines.", "Processing MHT File", MessageBoxButtons.OK, MessageBoxIcon.None, MessageBoxDefaultButton.Button1);
>
>                                       //string[] lines = (string[])(Lines.ToArray(typeof(string)));
>
>                                       if (filetype.Equals("xml"))
>                                       {
>                                               for (int i=0; i<Lines.Count; i++)
>                                               {
>                                                       string lin=(string)(Lines[i]);
>                                                       if (Regex.Match(lin,"BoundingBox").Success)
>                                                       {
>                                                               string tempwdl=wdl.SavedFilePath;
>                                                               //MessageBox.Show(lin , "Processing xml File",
MessageBoxButtons.OK, MessageBoxIcon.None, MessageBoxDefaultButton.Button1);
>                                                               string[] temp = lin.Split("");
>                                                               wdl.North = decimal.Parse(temp[3]);
>                                                               wdl.West = decimal.Parse(temp[5]);
>                                                               wdl.South = decimal.Parse(temp[7]);
>                                                               wdl.East = decimal.Parse(temp[9]);
>                                                               wdl.Name = temp[11]; //The date stamp
>                                                               string path =
Path.GetDirectoryName(wdl.SavedFilePath);
>                                                               wdl.SavedFilePath= Path.Combine( path,
wdl.Name.Replace(":",""));
>                                                               //Need to somehow create a copy of the modified wdl
for adding to the array list.
>                                                               FramesData.Add(new WMSDownload(wdl));
>                                                               //FramesData.Add(wdl);
>                                                               //MessageBox.Show(wdl.SavedFilePath, "Naming
Frame Files", MessageBoxButtons.OK, MessageBoxIcon.None, MessageBoxDefaultButton.Button1);
>                                                               wdl.SavedFilePath=tempwdl;
>
>                                                       }
>                                               }
>                                       }
>                                       if (filetype.Equals("AVI"))
```

```
>                                       {
>                                               MessageBox.Show(wdl.SavedFilePath, "Writing AVI File",
MessageBoxButtons.OK, MessageBoxIcon.None, MessageBoxDefaultButton.Button1);
>                                               imgend=(int)(br.BaseStream.Position);
>                                               int imgsize = imgend-imgstart;
>                                               byte[] img = new byte[imgsize];
>                                               br.BaseStream.Seek((long)(imgstart+2),SeekOrigin.Begin);
>                                               br.Read(img,0,img.Length);
>                                               string path = Path.GetDirectoryName(wdl.SavedFilePath);
>                                               string moviename = Path.Combine(path,"movie.avi");
>                                               FileStream fs = new FileStream(moviename,FileMode.OpenOrCreate);
>                                               BinaryWriter bw = new BinaryWriter(fs);
>                                               bw.Write(img,0,img.Length);
>                                               bw.Close();
>                                       }
>                                       Lines.Clear();
>                                       filetype="";
>                                       line=ReadBinLine(br);
>                               }
>                               br.Close();
>                               //MessageBox.Show("Completed Layer Extraction", "Processing MHT File",
MessageBoxButtons.OK, MessageBoxIcon.None, MessageBoxDefaultButton.Button1);
>                               //Here we must split the AVI into frame files I don't know how yet.
>                               //This is a hack using mplayer to split the movie into PNG files.
>                               string pathmovie = Path.Combine(Path.GetDirectoryName(wdl.SavedFilePath),"movieframes");
>                               string currentd = Directory.GetCurrentDirectory();
>                               //MessageBox.Show(pathmovie, "Processing AVI File", MessageBoxButtons.OK,
MessageBoxIcon.None, MessageBoxDefaultButton.Button1);
>                               try {
>                               Directory.CreateDirectory(pathmovie);
>                               Environment.CurrentDirectory=pathmovie;
>                               System.Diagnostics.Process proc = new System.Diagnostics.Process();
>                               proc.EnableRaisingEvents=false;
>                               proc.StartInfo.FileName="c:/mplayer/mplayer";
>                               proc.StartInfo.Arguments="-vo png
\""+Path.Combine(Path.GetDirectoryName(wdl.SavedFilePath),"movie.avi")+"\"";
>                               //MessageBox.Show(proc.StartInfo.FileName+proc.StartInfo.Arguments, "Spliting AVI to PNG",
MessageBoxButtons.OK, MessageBoxIcon.None, MessageBoxDefaultButton.Button1);
>                               proc.StartInfo.UseShellExecute=true;
>                               proc.Start();
>
>                               proc.WaitForExit();
>                               Environment.CurrentDirectory=currentd;
>                               } catch (Exception E)
>                               {
>                                       MessageBox.Show(E.Message, "Error Processing AVI File", MessageBoxButtons.OK,
MessageBoxIcon.None, MessageBoxDefaultButton.Button1);
>
>                               }
>                               //Now we have the png files that make up the movie in the movie subfolder.
>                               string[] frames=Directory.GetFiles(pathmovie);
>
>                               Environment.CurrentDirectory=currentd;
>                               int framecount=0;
>                               foreach (string image in frames)
>                               {
>                                       //if ((Regex.Match(image,"$png").Success)||(Regex.Match(image,"$PNG").Success))
>                                       //{
>                                       WMSDownload thisframe = (WMSDownload)(FramesData[framecount]);
```

```
>                                       //MessageBox.Show(thisframe.SavedFilePath, "Processing Image Files",
MessageBoxButtons.OK, MessageBoxIcon.None, MessageBoxDefaultButton.Button1);
>
>                                       //thisframe.SavedFilePath = image;
>                                       Stream datastream = new FileStream(image,FileMode.Open);
>                                       ImageHelper.ConvertToDxt3(datastream, thisframe.SavedFilePath+".dds",
worldWindow.DrawArgs.device );
>                                       if(this.animationState == AnimationState.Cancel || animationState ==
AnimationState.Stop)
>                                               return;
>                                       ImageLayerInfo imageLayerInfo = new ImageLayerInfo(thisframe);
>                                       imageLayerInfo.Save(thisframe.SavedFilePath + ".wwi");
>                                       lock(this.animationFrames.SyncRoot)
>                                               animationFrames.Add(imageLayerInfo);
>                                               animationLayers.Add(null);
>                                       framecount++;
>                                       //}
>                               }
>
>                       }
>
>
> /*                    private void ProcessDownloadedFrames(WMSDownload wdl, WebDownload dl)
>                       {
>                               BinaryReader br = new BinaryReader(dl.ContentStream);
>                               string seperator=dl.ContentType.Substring(25);
>                               string line = ReadBinLine(br);
>                               string[] filenames = wdl.Date.Split(',');
>                               while (line != "--"+seperator)
>                               {
>                                       //MessageBox.Show(line, "Process Header Line", MessageBoxButtons.OK,
MessageBoxIcon.None, MessageBoxDefaultButton.Button1);
>                                       line = ReadBinLine(br);
>                               }
>                               foreach(string filename in filenames)
>                               {
>                                       string path = Path.GetDirectoryName(wdl.SavedFilePath);
>                                       if(filename!=null && filename.Length>0)
>                                       {
>                                               wdl.SavedFilePath = Path.Combine( path, filename.Replace(":",""));
>                                               wdl.Title += "\n" + filename;
>                                       }
>                                       else
>                                               wdl.SavedFilePath = Path.Combine( path, "Default" );
>                                       line=ReadBinLine(br);
>                                       while (line != "--"+seperator)
>                                       {
>                                               line = ReadBinLine(br);
>                                               if (line.Length > seperator.Length+4)
>                                               {
>                                                       //MessageBox.Show(line, "Process xml Line",
MessageBoxButtons.OK, MessageBoxIcon.None, MessageBoxDefaultButton.Button1);
>                                                       //<BoundingBox SRS="EPSG:4326" minx="  46.3853370 "
miny="  -122.3607050" maxx="  46.1214770 " maxy=" -121.9799050" />
>                                                       string[] temp = line.Split('"');
>
>                                                       wdl.North = decimal.Parse(temp[3]);
>                                                       wdl.West = decimal.Parse(temp[5]);
>                                                       wdl.South = decimal.Parse(temp[7]);
>                                                       wdl.East = decimal.Parse(temp[9]);
```

```
>                                                       //MessageBox.Show("North="+wdl.North.ToString()+"
West="+wdl.West.ToString()+" South="+wdl.South.ToString()+" East="+wdl.East.ToString(), "Bounding Box",
MessageBoxButtons.OK, MessageBoxIcon.None, MessageBoxDefaultButton.Button1);
>                                                       }
>                                               }
>                                               line = ReadBinLine(br); //Read and discard content-type header.
>                                               line = ReadBinLine(br); //discard extra CRLF.
>                                               MessageBox.Show("Creating Ouput Stream", "Process Image Lines",
MessageBoxButtons.OK, MessageBoxIcon.None, MessageBoxDefaultButton.Button1);
>                                               Directory.CreateDirectory(Path.GetDirectoryName(wdl.SavedFilePath));
>                                               string ddsimage = wdl.SavedFilePath + ".png";
>                                               Stream ds = new FileStream(ddsimage,FileMode.Create);
>                                               MessageBox.Show("Ouput Stream Created", "Process Image Lines",
MessageBoxButtons.OK, MessageBoxIcon.None, MessageBoxDefaultButton.Button1);
>                                               int position=(int)(br.BaseStream.Position);
>                                               UpdateStatusBar("DownLoading Frame...");
>                                               BinaryWriter bw = new BinaryWriter(ds);
>                                               //string lastline = "\n\r--"+seperator+"--\n\r";
>                                               bool done=false;
>                                               int count=0;
>                                               while (!(done))
>                                               {
>                                                       char[] test = ("--"+seperator).ToCharArray();
>                                                       char[] final;
>                                                       if  (br.PeekChar()!='-')
>                                                       {
>                                                               bw.Write(br.ReadChar());
>                                                       }
>                                                       else
>                                                       {
>                                                               final=br.ReadChars(test.Length);
>                                                               string test2="";
>                                                               foreach (char c2 in final)
>                                                               {
>                                                                       test2=test2+c2;
>                                                               }
>                                                               MessageBox.Show("Testing for end of image marker.\n\r"+test2,
"Process Image Lines", MessageBoxButtons.OK, MessageBoxIcon.None, MessageBoxDefaultButton.Button1);
>                                                               if (test2=="--"+seperator)
>                                                               {
>                                                                       done=true;
>                                                               }
>                                                               else
>                                                               {
>                                                                       bw.Write(final);
>                                                                       final=null;
>                                                               }
>                                                       }
>                                                       count++;
>                                                       if (count > 1000)
>                                                       {
>                                                               MessageBox.Show("The Image Won't end!!!", "I am stuck here",
MessageBoxButtons.OK, MessageBoxIcon.None, MessageBoxDefaultButton.Button1);
>                                                               count=0;
>                                                       }
>                                               }
>                                               MessageBox.Show("Done Processing Image", "Processed Image Lines",
MessageBoxButtons.OK, MessageBoxIcon.None, MessageBoxDefaultButton.Button1);
>                                               bw.Close();
>                                               ds.Close();
```

```
>                                       Stream datastream = new FileStream(ddsimage,FileMode.Open);
>                                       try
>                                       {
>                                               string ddsFile = wdl.SavedFilePath + ".dds";
>                                               if( wdl.Date.Length>0)
>                                                       UpdateStatusBar( "Converting " + wdl.Date );
>                                               else
>                                                       UpdateStatusBar( "Converting... " );
>
        Directory.CreateDirectory(Path.GetDirectoryName(wdl.SavedFilePath));
>                                                       if (datastream.Length==0)
>                                                               throw new WebException("Server returned no data.");
>                                                       ImageHelper.ConvertToDxt3(datastream, ddsFile,
worldWindow.DrawArgs.device );
>                                               if(this.animationState == AnimationState.Cancel || animationState ==
AnimationState.Stop)
>                                                       return;
>                                               ImageLayerInfo imageLayerInfo = new ImageLayerInfo(wdl);
>                                               imageLayerInfo.Save(wdl.SavedFilePath + ".wwi");
>                                               lock(this.animationFrames.SyncRoot)
>                                               animationFrames.Add(imageLayerInfo);
>                                               animationLayers.Add(null);
>                                               if(this.animationState == AnimationState.Cancel || this.animationState ==
AnimationState.Stop)
>                                                       return;
>
>                                               UpdateStatusBar( "" );
>                                       }
>                                       catch(Exception caught)
>                                       {
>                                               this.worldWindow.Caption = caught.Message;
>                                               Utility.Log.Write( caught );
>                                               MessageBox.Show(caught.ToString(), "Frame Image Extraction Error",
MessageBoxButtons.OK, MessageBoxIcon.None, MessageBoxDefaultButton.Button1);
>                                       }
>                                       datastream.Close();
>                               }
>                               br.Close();
1670c2306,2307
<
---
>                       */
>
2082a2720,2731
>
>                       private void EnqueueFrames(WMSDownload dl)
>                       {
>                               this.frameQueue.Enqueue(dl);
>                               StartFramesThread();
>                       }
>
>                       private void EnqueueMovie(WMSDownload dl)
>                       {
>                               this.movieQueue.Enqueue(dl);
>                               StartMovieThread();
>                       }
2183c2832
<                               Cancel
---
>                               Cancel
```

```
2308a2958,3034
>
>
>                       void GetMultipleFramesBtnClick(object sender, System.EventArgs e)
>                       {
>                               if(this.animationState == AnimationState.Play)
>                               {
>                                       animationState = AnimationState.Pause;
>                                       buttonPlay.ImageIndex = 4;
>                                       return;
>                               }
>
>                               if(this.animationState == AnimationState.Pause)
>                               {
>                                       animationState = AnimationState.Play;
>                                       buttonPlay.ImageIndex = 2;
>                                       animationTimer.Start();
>                                       return;
>                               }
>
>                               if(this.animationState == AnimationState.Stop)
>                               {
>                                       MyWMSLayer curLayer = this.getCurrentlySelectedWMSLayer();
>                                       if(curLayer == null)
>                                               return;
>
>                                       WMSLayerStyle curStyle = this.getCurrentlySelectedWMSLayerStyle();
>                                       if(curLayer.Dates == null)
>                                               return;
>
>                                       this.curLoadedLayer = curLayer;
>                                       this.curLoadedStyle = curStyle;
>
>                                       buttonLegend.Enabled = curLayer.HasLegend;
>
>
>                                       if(this.comboBoxAnimationStartTime.SelectedIndex == -1 ||
>                                               this.comboBoxAnimationEndTime.SelectedIndex == -1)
>                                               return;
>
>                                       if(this.comboBoxAnimationStartTime.SelectedIndex >
this.comboBoxAnimationEndTime.SelectedIndex)
>                                       {
>                                               // Swap start and end
>                                               int start = comboBoxAnimationEndTime.SelectedIndex;
>                                               comboBoxAnimationEndTime.SelectedIndex =
comboBoxAnimationStartTime.SelectedIndex;
>                                               comboBoxAnimationStartTime.SelectedIndex = start;
>                                       }
>
>
>                                       if(this.treeViewTableOfContents.SelectedNode.Tag == null)
>                                               //wms layer not properly selected
>                                               return;
>
>                                       this.Reset();
>
>                                       this.groupBoxAnimationTimeFrame.Enabled = false;
>                                       this.groupBoxExtents.Enabled = false;
>                                       string dates="";
```

```
>                                       for(int i = this.comboBoxAnimationStartTime.SelectedIndex; i <
this.comboBoxAnimationEndTime.SelectedIndex; i++)
>                                       {
>                                               dates=dates+curLayer.Dates[i]+",";
>                                       }
>                                       dates=dates+curLayer.Dates[this.comboBoxAnimationEndTime.SelectedIndex];
>                                       bool allframes=false;
>                                       if((this.comboBoxAnimationStartTime.SelectedIndex ==
0)&&(this.comboBoxAnimationEndTime.SelectedIndex == this.comboBoxAnimationEndTime.Items.Count-1))
>                                               allframes=true;
>                                       WMSDownload wmsDownload = curLayer.GetWmsFrames(
>                                                       dates,
>                                                       curStyle,
>                                                       CacheDirectory,allframes);
>                                       EnqueueFrames(wmsDownload);
>                                       //EnqueueDownload(wmsDownload);
>                                       buttonPlay.ImageIndex = 2;
>                                       animationState = AnimationState.Play;
>                                       animationTimer.Start();
>                               }
>
>                       }
2309a3036,3086
>
>                       void GetMovieBtnClick(object sender, System.EventArgs e)
>                       {
>                               if(this.animationState == AnimationState.Play)
>                               {
>                                       animationState = AnimationState.Pause;
>                                       buttonPlay.ImageIndex = 4;
>                                       return;
>                               }
>
>                               if(this.animationState == AnimationState.Pause)
>                               {
>                                       animationState = AnimationState.Play;
>                                       buttonPlay.ImageIndex = 2;
>                                       animationTimer.Start();
>                                       return;
>                               }
>
>                               if(this.animationState == AnimationState.Stop)
>                               {
>                                       MyWMSLayer curLayer = this.getCurrentlySelectedWMSLayer();
>                                       if(curLayer == null)
>                                               return;
>
>                                       WMSLayerStyle curStyle = this.getCurrentlySelectedWMSLayerStyle();
>                                       if(curLayer.Dates == null)
>                                               return;
>
>                                       this.curLoadedLayer = curLayer;
>                                       this.curLoadedStyle = curStyle;
>
>                                       buttonLegend.Enabled = curLayer.HasLegend;
>
>                                       if(this.treeViewTableOfContents.SelectedNode.Tag == null)
>                                               //wms layer not properly selected
>                                               return;
>
```

```
>                                       this.Reset();
>
>                                       this.groupBoxAnimationTimeFrame.Enabled = false;
>                                       this.groupBoxExtents.Enabled = false;
>                                       WMSDownload wmsDownload = curLayer.GetWMSMovie(
>                                                       curStyle,
>                                                       CacheDirectory);
>                                       EnqueueMovie(wmsDownload);
>                                       //EnqueueDownload(wmsDownload);
>                                       buttonPlay.ImageIndex = 2;
>                                       animationState = AnimationState.Play;
>                                       animationTimer.Start();
>                               }
>                       }
2905c3682
<
---
>
2966a3744,3888
>
>                       public WMSDownload GetWMSMovie(WMSLayerStyle curStyle, string cacheDirectory)
>                       {
>                               string url="";
>                               url = GetWMSMovieURL((curStyle != null ? curStyle.name : null));
>                               WMSDownload wmsDownload = new WMSDownload( url );
>                               wmsDownload.Title = this._title;
>                               string path = Path.Combine( cacheDirectory , Path.Combine(
>                                       this._parentWMSList.Name,
>                                       this._name + (curStyle != null ? curStyle.name : "")));
>                               wmsDownload.SavedFilePath = Path.Combine( path,"Movie");
>                               return wmsDownload;
>                       }
>
>                       public string GetWMSMovieURL(string style)
>                       {
>                               if(this._name == null)
>                               {
>                                       Utility.Log.Write("WMSB", "No Name");
>                                       return null;
>                               }
>                               string projectionRequest = "";
>                               if(this.ParentWMSList.Version == "1.1.1")
>                                       projectionRequest = "&srs=EPSG:4326";
>                               else if(this.ParentWMSList.Version == "1.3.0")
>                                       projectionRequest = "&crs=CRS:84";
>
>                               string imageFormat = null;
>
>                               if(this._imageFormats == null)
>                               {
>                                       Utility.Log.Write("WMSB", "No formats");
>                                       return null;
>                               }
>
>                               foreach(string curFormat in this._imageFormats)
>                               {
>                                       if(string.Compare(curFormat, "image/png", true, CultureInfo.InvariantCulture) == 0)
>                                       {
>                                               imageFormat = curFormat;
>                                               break;
```

```
>                                       }
>                                       if(string.Compare(curFormat, "image/jpeg", true, CultureInfo.InvariantCulture) == 0 ||
>                                               String.Compare(curFormat, "image/jpg", true, CultureInfo.InvariantCulture)
== 0)
>                                       {
>                                               imageFormat = curFormat;
>                                       }
>                               }
>                               if(imageFormat == null)
>                                       return null;
>
>                               string wmsQuery = string.Format(
>                                       CultureInfo.InvariantCulture,
>
        "{0}?service=WMS&version={1}&request=GetMovie&layers={2}&format={3}&time=&styles={4}",
>                                       this.ParentWMSList.ServerGetMapUrl,
>                                       this.ParentWMSList.Version,
>                                       this._name,
>                                       imageFormat,
>                                       projectionRequest,
>                                       (style != null ? style : ""));
>                               return wmsQuery;
>                       }
>
>                       public WMSDownload GetWmsFrames( string dateString,
>                                       WMSLayerStyle curStyle,
>                                       string cacheDirectory,bool allframes)
>                       {
>                               string url="";
>                               if (allframes)
>                               {
>                                       url = GetWMSFramesUrl (null, (curStyle != null ? curStyle.name : null));
>                               } else {
>                                       url = GetWMSFramesUrl (dateString, (curStyle != null ? curStyle.name : null));
>                               }
>                               WMSDownload wmsDownload = new WMSDownload( url );
>                               wmsDownload.Title = this._title;
>                               wmsDownload.Date = dateString;
>                               if(curStyle != null)
>                                       wmsDownload.Title += " (" + curStyle.title + ")";
>
>                               string path = Path.Combine( cacheDirectory , Path.Combine(
>                                       this._parentWMSList.Name,
>                                       this._name + (curStyle != null ? curStyle.name : "")));
>                               if(dateString!=null && dateString.Length>0)
>                               {
>                                       string[] dates=dateString.Split(',');
>                                       wmsDownload.SavedFilePath = Path.Combine( path, dates[0].Replace(":",""));
>                                       wmsDownload.Title += "\n" + dateString;
>                               }
>                               else
>                                       wmsDownload.SavedFilePath = Path.Combine( path, "Default" );
>                               return wmsDownload;
>                       }
>
>                       public string GetWMSFramesUrl(string date, string style)
>                       {
>                               if(this._name == null)
>                               {
>                                       Utility.Log.Write("WMSB", "No Name");
```

```
>                                       return null;
>                                   }
>                                   string projectionRequest = "";
>                                   if(this.ParentWMSList.Version == "1.1.1")
>                                       projectionRequest = "&srs=EPSG:4326";
>                                   else if(this.ParentWMSList.Version == "1.3.0")
>                                       projectionRequest = "&crs=CRS:84";
>
>                                   string imageFormat = null;
>
>                                   if(this._imageFormats == null)
>                                   {
>                                       Utility.Log.Write("WMSB", "No formats");
>                                       return null;
>                                   }
>
>                                   foreach(string curFormat in this._imageFormats)
>                                   {
>                                       if(string.Compare(curFormat, "image/png", true, CultureInfo.InvariantCulture) == 0)
>                                       {
>                                           imageFormat = curFormat;
>                                           break;
>                                       }
>                                       if(string.Compare(curFormat, "image/jpeg", true, CultureInfo.InvariantCulture) == 0 ||
>                                           String.Compare(curFormat, "image/jpg", true, CultureInfo.InvariantCulture)
== 0)
>                                       {
>                                           imageFormat = curFormat;
>                                       }
>                                   }
>
>                                   if(imageFormat == null)
>                                       return null;
>
>                                   string wmsQuery = string.Format(
>                                       CultureInfo.InvariantCulture,
>
            "{0}?service=WMS&version={1}&request=GetFrames&layers={2}&format={3}&time={4}{5}&styles={6}",
>                                       this.ParentWMSList.ServerGetMapUrl,
>                                       this.ParentWMSList.Version,
>                                       this._name,
>                                       imageFormat,
>                                       (date != null ? date : ""),
>                                       projectionRequest,
>                                       (style != null ? style : ""));
>                                   return wmsQuery;
>                               }
>
```