

WCS Server Use Cases

Explains how to use MapServer WCS with Modis, Landsat, DEM and NetCDF data.

MapServer WCS Use Cases

This document explains how to use MapServer to deliver Landsat, SPOT, DEM, and NetCDF temporal/banded data through the MapServer WCS interface. Thanks go to Steve Lime and Frank Warmerdam for their assistance with these projects

Landsat

To serve Landsat imagery through the MapServer Web Coverage Service specify the OutputFormat Object. For format support install the GDAL library and from the Command Prompt cd to where GDAL is installed and use the command, `gdalinfo --formats`. A list of all supported formats will appear and will specify if the format is read only `<ro>` or read and write `<rw>` for WCS the format needs to be supported for read and write.

For the example below the Landsat 7 15m resolution mosaic is in a Enhanced Compressed Wavelets format (ECW). By running the `gdalinfo.exe` program I could verify that the ECW format has write permissions, therefore the format can be specified in the MapFile and requested using the GetCoverage request.

```
OUTPUTFORMAT
  NAME ECW
  DRIVER "GDAL/ECW"
  MIMETYPE "image/ecw"
  IMAGEMODE "BYTE"
  EXTENSION "ecw"
END
```

The keyword "DUMP" must be set to true to allow the data to download during a GetCoverage request.

```
LAYER
  NAME "Landsat7"
  STATUS OFF
  TYPE RASTER
  PROCESSING "SCALE=AUTO"
  UNITS Meters
  DUMP TRUE
  TILEINDEX "MapServer/wcs/landsat7/l7mosaic15m.shp"
  TILEITEM "Location"
  METADATA
    "wcs_description" "Landsat 7 15m resolution mosaic"
    "wcs_name" "Landsat7"
    "wcs_label" "Landsat 7 15m resolution mosaic"
    "ows_srs" "EPSG:27700"
    "ows_extent" "0 0 700005 1050000"
    "wcs_resolution" "75 75"
    "wcs_bandcount" "3"
    "wcs_formats" "ECW"
  END
END
```

A GetCoverage request can then be requested (using the parameters set in the MapFile) by creating a URL with the elements: - Your Server, MapServer Program, Location of MapFile, Type of Service (WCS), Request (GetCoverage), Coverage (Landsat7), BBOX (0,0,700005,1050000), CRS (EPSG:27700), ResX (75) ResY (75), Format (ECW).

SPOT

SPOT imagery can be delivered through MapServer Web Coverage Service similarly to the Landsat example above. The main difference is that as SPOT is a greyscale image the `wcs_bandcount = 1` rather than a Landsat image which consists of 3 bands. For this example the well known GeoTiff format will be used to demonstrate what to specify in a MapFile for SPOT data.

```
OUTPUTFORMAT
  NAME GEOTIFF
  DRIVER "GDAL/GTiff"
  MIMETYPE "image/tiff"
  IMAGEMODE "BYTE"
  EXTENSION "tif"
END
LAYER
  NAME "SPOT"
  STATUS OFF
  TYPE RASTER
  PROCESSING "SCALE=AUTO"
  UNITS Meters
  DUMP TRUE
  TILEINDEX "MapServer/wcs/orthospot/spot.shp"
  TILEITEM "Location"
  METADATA
    "wcs_description" "Orthospot mosaic"
    "wcs_name" "SPOT"
    "wcs_label" "Orthospot mosaic"
    "ows_srs" "EPSG:27700"
    "ows_extent" "375960 64480 497410 200590"
    "wcs_resolution" "100 100"
    "wcs_bandcount" "1"
    "wcs_formats" "GEOTIFF"
    "wcs_nativeformat" "8-bit GeoTIF"
  END
END
```

The key parameters to specify in the WCS MapFile for any data layer and format are:

- Layer Name = Create a short name for the data
- Layer Type = Raster
- Layer Dump = True

The following examples further demonstrate how WCS can be implemented and also how to create WCS containing layers with a temporal dimension (see NetCDF example).

DEM

It is possible to deliver 16 bit DEM data through the MapServer Web Coverage Service.

Firstly it is necessary to specify the output format in the map file:

```
OUTPUTFORMAT
  NAME GEOTIFFINT16
  DRIVER "GDAL/GTiff"
  MIMETYPE "image/tiff"
  IMAGEMODE "INT16"
  EXTENSION "tif"
END
```

and in the corresponding layer to specify the keyword "DUMP" to be true:

```
LAYER
  NAME "srtm"
  STATUS OFF
  TYPE RASTER
  DUMP TRUE
  DATA "srtm.tif"
  PROJECTION
    "init=epsg:4326"
  END
  METADATA
    wcs_label "SRTM WCS TIF Server"
    ows_extent "-180 -90 180 90"
    wcs_resolution "0.00083 -0.00083"
    ows_srs "EPSG:4326"
    wcs_formats "GEOTIFFINT16"
    wcs_nativeformat "geotiff"
  END
END
```

Performance gains can be made by using the gdaladdo utility described at http://www.remotesensing.org/gdal/gdal_utilities.html#gdaladdo

NetCDF

Firstly GDAL doesn't support all versions of netCDF (there are a lot, it is a generic format), so for stability it may be necessary to convert the files into GeoTiff format first. This can be achieved using the netCDF libraries here <http://my.unidata.ucar.edu/content/software/netcdf/index.html>. Denis Nadeau and Frank Warmerdam have added netCDF CF as a read only format within GDAL, so it is now possible to read the CF convention netCDF files directly from disk.

We placed the Z-levels in the bands of the GDAL data file (either GeoTiff or netCDF), and created a shape index for the time levels. GDAL data is a 2-D format (x,y) and bands. netCDF is an N-D file format, supporting time, x,y,z, and experiment parameters. By using a set of GDAL netCDF / geoTiff files it is possible to represent this, and to store the z-level (height) as bands within the data file. Although a hack, it is possible for a custom client to receive important metadata from the describeCoverage operation of a WCS about the which z-level a band of a geotiff represents by encoding this in the returned axes description tag.

To create the shape file for the temporal dimension we had to do some hacking with Java code, but we also got it to work with Steve Lime's perl script in the MODIS MapServer demo download (which doesn't seem to be available now).

The perl script used in Modis demo by Steve Lime is as follows, and I have placed inline comments below. The script assumes that gdaltindex has already been run in this directory to create a tile index shape and dbf file. It assumes that the filenames of your data files have the date in the filename, for example myfileYYYYMMDDHH.tif

```
#!/usr/bin/perl
use XBase;
opendir(DIR, '.'); # open the current directory
foreach $file (readdir(DIR)) {
  next if !($file =~ /\.dbf$/); # read the dbf file in this directory created by gdaltindex
  print "Working on $file...\n";
  $tfile = 'temporary.dbf';
  system("mv $file $tfile");
  $oldtable = new XBase $tfile or die XBase->errstr;
```

```

print join("\t", $oldtable->field_names) ."\n";
print join("\t", $oldtable->field_types) ."\n";
print join("\t", $oldtable->field_lengths) ."\n";
print join("\t", $oldtable->field_decimals) ."\n";
$newtable = XBase->create("name" => $file,
    "field_names" => [$oldtable->field_names, "IMGDATE"], # this is the FILTERITEM i
    "field_types" => [$oldtable->field_types, "C"], # character column type
    "field_lengths" => [$oldtable->field_lengths, 13], # length of the date string
    "field_decimals" => [$oldtable->field_decimals, undef]) or die "Error creating ne
foreach (0 .. $oldtable->last_record) {
($deleted, @data) = $oldtable->get_record($_);
print " ...record $data[0]\n";
    # extract the date
    $year = substr $data[0], 8, 4; # year is at position 8 in the filename string
    $month = substr $data[0], 12, 2; # month is at position 12 in the filename string
    $day = substr $data[0], 14, 2; # day is at position 14 in the filename string
    $hour = substr $data[0], 16, 2; # hour is at position 16 in the filename string
    $date = "$year-$month-$day" . "T" . "$hour\n"; # format is YYYY-MM-DDTHH, or any ISO forma
    print "$date";
    push @data, "$date";
$newtable->set_record($_, @data);
}
$newtable->close();
$oldtable->close();
unlink($tfile);
}

```

If have used the perl script then skip to the layer definitions below, if you wish to code your own the description is here.

The DBF file has to have the column 'location' that indicates the location of the data file (either absolute path or relative to the map file location, and the second column that can be called whatever you want but indexes time. In our case we called it 'time' :-)

The corresponding shapefile then has to contain Polygons with the bounding boxes of the tif file for each time. So OGRInfo timeIndex.shp looks something like:

```

OGRFeature(timeIndex):116
  location(String) = mytime.tif
  time(String) = 2001-01-31T18:00:00
  POLYGON ((xxx,xxxx,.....))

```

Define your output format as:

```

OUTPUTFORMAT
NAME GEOTIFF_FLOAT
DRIVER 'GDAL/GTiff'
MIMETYPE 'image/tiff'
IMAGEMODE FLOAT32
EXTENSION 'tif'
END

```

Then you need to define your tile index within the map file:

```

LAYER
NAME 'time_idx'
TYPE TILEINDEX
DATA 'timeIndex'
FILTERITEM 'time'
FILTER '%time%'
END

```

and the actual layer:

```
LAYER
  NAME 'TempData'
  STATUS OFF
  TYPE RASTER
  DUMP TRUE
  TILEINDEX 'time_idx'
  PROJECTION
    "init=epsg:4326"
  END
  METADATA
    wcs_label 'Temperature data'
    ows_extent '-180 -90 180 90'
    wcs_resolution '1.125 -1.125'
    ows_srs 'EPSG:4326'
    wcs_formats 'GEOTIFF_FLOAT'
    wcs_nativeformat 'netCDF'
    wcs_bandcount '27'
    wcs_rangeset_axes 'bands'
    wcs_rangeset_label 'Pressure (hPa units) Levels'
    wcs_rangeset_name 'bands'
    wcs_rangeset_description 'Z levels '
    wcs_timeposition '2001-01-01T06:00:00,2001-01-01T12:00:00,2001-01-01T18:00:00,2001-01-02T00:00:00'
    wcs_timeitem 'time'
  END
END
```

The TempData coverage layer will now let you subset with the &bands=... &time=... subset parameters!

To do a coordinate reprojection specify in the request &Response_CRS=ESPG:xxxx

When you start doing temporal subsetting with WCS and MapServer you can see the need for an automatic way of generating map files such as using an XSL stylesheet!