Replacing MapServer with MediaFlux

# Background

James Cook University currently uses MapServer to deliver map tiles to mapping software running inside client web browsers. An example of this is the CliMAS project:

<http://climas.hpc.jcu.edu.au/maps/>

(select a species name, e.g. “cane toad”).

Tiles are derived from selected GeoTIFF files. Generally one GeoTIFF provides many tiles that are displayed as a single map layer. For example, the current distribution of cane toads across Australia is represented as a single GeoTIFF, but delivered to the browser as a dozen or so map tiles.

JCU will store the source GeoTIFF files in MediaFlux, and wants MediaFlux to be able to derive and deliver map tiles from GeoTIFF assets, so that MapServer can be removed from the technology stack for map displaying projects.

JCU anticipates that duplicating all MapServer functionality, or a full implementation of a mapping standard like WMS, would be a large undertaking, so this document describes a minimal viable product for MapServer replacement.

|  |  |  |
| --- | --- | --- |
| **Date** | **Author** | **Modification** |
| 31 March 2015 | Daniel.Baird@jcu.edu.au | Original version |
|  |  |  |

# Configuring tile production

The GeoTIFFs JCU intends to serve from MediaFlux are not genuine images. Each pixel value is not a colour; it is a data value.

For example, in a GeoTIFF of cane toad climate suitability, each pixel value varies from 0 to 1. The value represents the likelihood of a cane toad thriving at that position. To display this visually, each data value needs to be mapped to a pixel colour.

To generate a map tile the server needs to:

* Find the subset of source “data pixels” that represents the area requested within the source GeoTIFF
* Reduce (depending on zoom level) the number of “data pixels” in the resulting subset by sampling or averaging across multiple source pixels
* Remap each “data pixel” value to a pixel colour / transparency.

The server configuration needs to support a cutoff value, and specification of the colour and transparency mapping. This should allow for two colour mapping modes. Values in the configuration should include one of two alternative colour+transparency scale options, and an optional cutoff value.

**A cutoff value**, so data values below the supplied cutoff value are mapped to an output pixel with full transparency.

A **relative colour scale** specified by two colour+transparency values. One pixel value is applied to the lowest data value in the source file (or the cutoff value, if a cutoff is specified), the other pixel value is applied to the highest data value in the source file, and the output pixel for other data values is interpolated by mapping between the two ends of the scale. This gives a colour scale that is relative to values in the source data.

A **fixed-value colour scale** specified by two pairs, each pair being a data value plus a colour+transparency value. A pixel value is mapped, and the source values are interpolated between those values. Values outside the given limits should be mapped as if they were at the nearest limit. This gives a colour+transparency scale relative to fixed values.

## Examples

Here are some example configurations expressed in JSON and using the CSS rgba() function to describe colour and transparency.

**Example 1:** Map the source file’s minimum value to light blue at 10% opacity; map the source file’s maximum value to darker blue at 90% opacity.

rainfallMapConfig = {

“scale”: {

“min”: “rgba(100, 150, 255, 0.1)”,

“max”: “rgba( 0, 20, 100, 0.9)”

}

}

In Example 1, a value roughly midway between the source file’s maximum and minimum values would be mapped to a colour close to rgba(50, 85, 177, 0.5).

**Example 2:** Transparent below 0.1; map 0.1 to a red colour at 50% opacity; map the source file’s maximum value to green at 90% opacity.

climateSuitabilityMapConfig = {

“scale”: {

“min”: “rgba(255, 50, 50, 0.5)”,

“max”: “rgba( 0, 255, 0, 0.9)”

},

“cutoff”: 0.1

}

In Example 2, a value <= 0.04 would be 100% transparent. A value midway between source min and source max, if it was > 0.04, would be mapped to rgba(127, 152, 25, 0.75).

**Example 3:** Map value <= 1 to a yellow at 50% opacity; map values >= 1000 to green at 100% opacity.

speciesDensityMapConfig = {

“scale”: {

“1”: “rgba(255, 255, 0, 0.5)”,

“1000”: “rgba( 0, 255, 0, 1.0)”

},

cutoff: 0

}

In Example 3, value 0 gets transparent; value 100 would be about rgba(25, 255, 0.55), and value 3000 would get rgba(0, 255, 0, 1.0).

# Framing the map tile

There are various standards for map services, but a partial implementation of WMS that just covers tile delivery will suffice.

WMS includes a “GetMap” service which is used to handle map tile requests. The WMS required fields are:

|  |  |  |
| --- | --- | --- |
| **Required Parameter** | **WMS Description** | **What MediaFlux could do** |
| service | Service name. Value is WMS. | Ignore, or error if not WMS |
| version | Service version. Value is one of 1.0.0, 1.1.0, 1.1.1, 1.3. | Ignore |
| request | Operation name. Value is GetMap. | Error if not GetMap |
| layers | Layers to display on map. […] | Require DEFAULT |
| styles | Styles in which layers are to be rendered. […] | Ignore |
| srs *or* crs | Spatial Reference System for map output. […] | Error if not EPSG:3857 |
| bbox | Bounding box for map extent. Value is minx,miny,maxx,maxy in units of the SRS. | Use for bounds of map tile |
| width | Width of map output, in pixels. | Use for width of map tile |
| height | Height of map output, in pixels. | Use for height of map tile |
| format | Format for the map output. See [WMS output formats](http://docs.geoserver.org/stable/en/user/services/wms/outputformats.html#wms-output-formats) for supported values. | Error if not image/png |
| transparent | *(optional, but interesting)* Values are true or false | JCU will always want transparency. Maybe ignore this field? |

**Example URL** (with newlines added):

http://mediaflux.jcu.edu.au/path/to/geotiff/whatever/

?SERVICE=WMS

&REQUEST=GetMap

&VERSION=1.1.1

&LAYERS=DEFAULT

&STYLES=

&FORMAT=image%2Fpng

&TRANSPARENT=true

&HEIGHT=256

&WIDTH=256

&SRS=EPSG%3A3857

&BBOX=15028131.257091932,-4383204.9499851465,15654303.392804096,-3757032.814272983