

MapServer and OGC Web Services

<http://ms-ogc-workshop.maptools.org/>

Outline

- Web Services / Interoperability
- Applying OGC Web Services with MapServer:
 - OGC:WMS
 - with OGC:SLD
 - OGC:WFS/OGC:GML
 - OGC:WCS
 - OGC:SOS
 - OGC:WMC
 - OGC:Filter
 - Brokering to distributed Web Services for Geocoding
- Future goodies / Nice-to-Haves / Issues

Workshop

- Homepage
 - <http://ms-ogc-workshop.maptools.org/>
- Download
 - <http://ms-ogc-workshop.maptools.org/dl/>
- CVS
 - CVSROOT
 - :pserver:cvsanon@cvs.maptools.org:/cvs/maptools/cvsroot
 - Module: ms_ogc_workshop
- Support
 - [mapserver-users](#) mailing list

Workshop Package

- Install
 - Extract [ms_ogc_workshop.zip](#) (WinZip)
 - Extract to root of drive letter of ms4w location (i.e. C:\)
 - Make sure “Use Folder Names” is checked
 - Extracts into existing ms4w directory structure
- Base Directory:
 - [\ms4w\apps\ms_ogc_workshop\](#)
- Startup:
 - Start Apache:
 - [\ms4w\Apache\bin\Apache.exe](#)
 - Go to: http://127.0.0.1/ms_ogc_workshop/index.html

Web Services / Interoperability

- Distributed data through services interface
 - Less redundant data
 - Authoritative
 - More effective data management
 - Connect rather than hoard
- Transparent / cooperative

Web Services / Interoperability

- Based on open specifications
 - W3C, OGC
- Normalizes playing field, independent of:
 - Operating System
 - Programming Languages
 - Development Environments
- How?
 - Web Services!

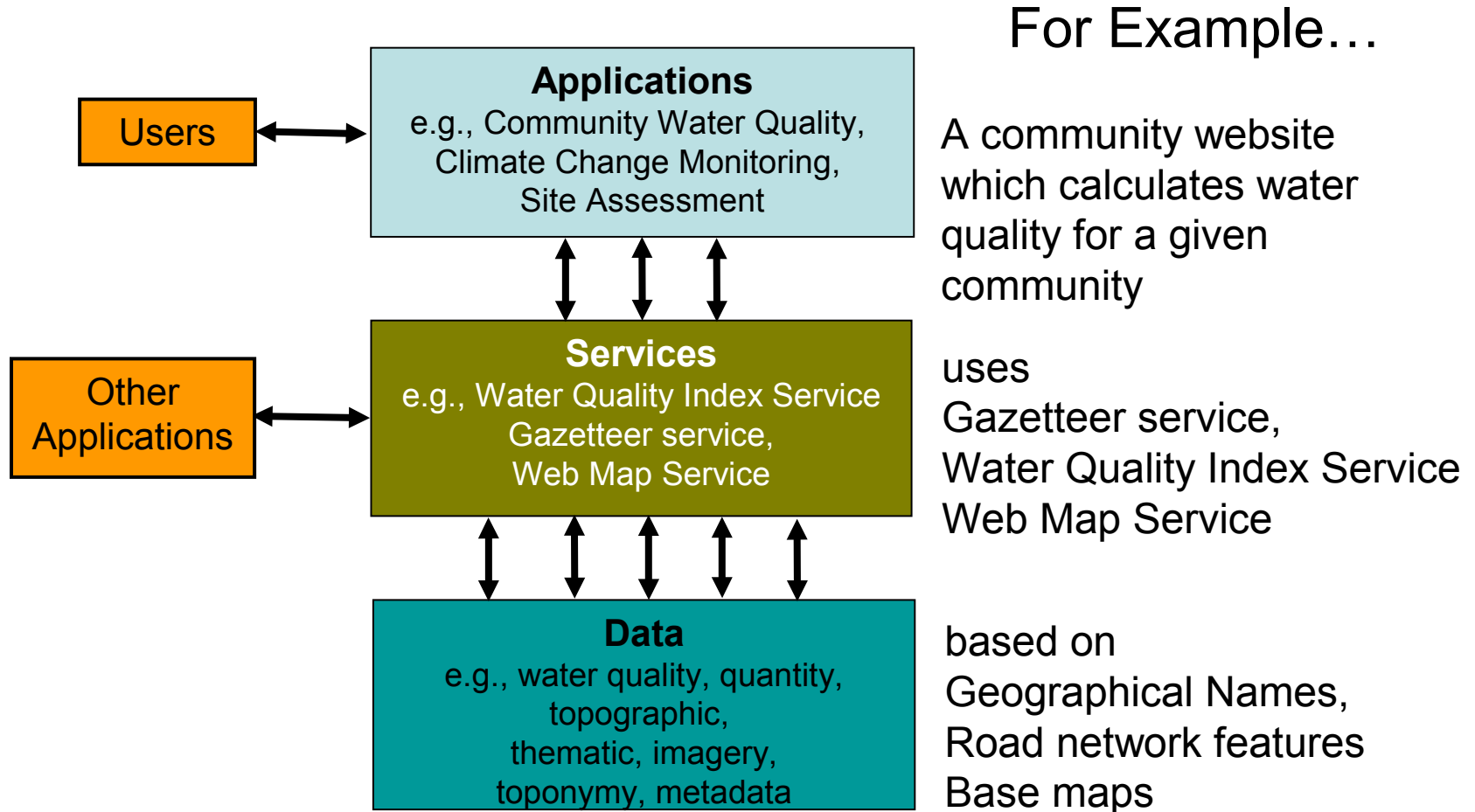
Geospatial Interoperability

- “*Geospatial Interoperability* is the ability for two different software systems to interact with geospatial information. Interoperability between heterogeneous computer systems is essential to providing geospatial data, maps, cartographic and decision support services, and analytical functions. Geospatial interoperability is dependent on voluntary, consensus-based standards... These *geospatial standards* are essential to advancing data access and collaborations in e-Government, natural hazards, weather and climate, exploration, and global earth observation.”
 - Geospatial Interoperability (GI) Return on Investment Study Report , NASA, April 2005
 - <http://gio.gsfc.nasa.gov/docs/ROI%20Study.pdf>

Web Services Defined

- Web Service = any software which makes itself available over the Internet and uses a standard XML messaging system
- XML makes this happen
- Provides more control to application developer (raw vs. refined goods)

Web Services Architecture Approach



Open Geospatial Consortium (OGC)

- Circa 1994
- Web Services for Geospatial Interoperability
- “Develop first, then spec” approach

OGC Specifications

- **WMS**
- **WFS**
- **WCS**
- **WMC**
- **GML**
- **SLD**
- **SOS, OM**
- **Filter**
- **Catalog**
- WCTS
- GO-1
- OWS Common
- Grid Coverages
- Location Services
- Simple Features
 - CORBA
 - SQL
 - OLE/COM

Bold = Supported by MapServer

MapServer and OWS

- Enabling OWS in MapServer
 - Build software with:
 - `--with-wms \`
 - `--with-wmsclient \`
 - `--with-wfs \`
 - `--with-wfsclient \`
 - `--with-wcs`
 - `--with-sos`
 - NB: dependent libraries needed
 - Read documentation
 - Get predefined Builds
 - Windows, MS4W, FGS

MapServer and OWS

- Enabling OWS in MapServer
 - Same old MapServer; just populate mapfiles with appropriate directives
 - MapServer OGC documentation HOWTOs

MapServer and OWS

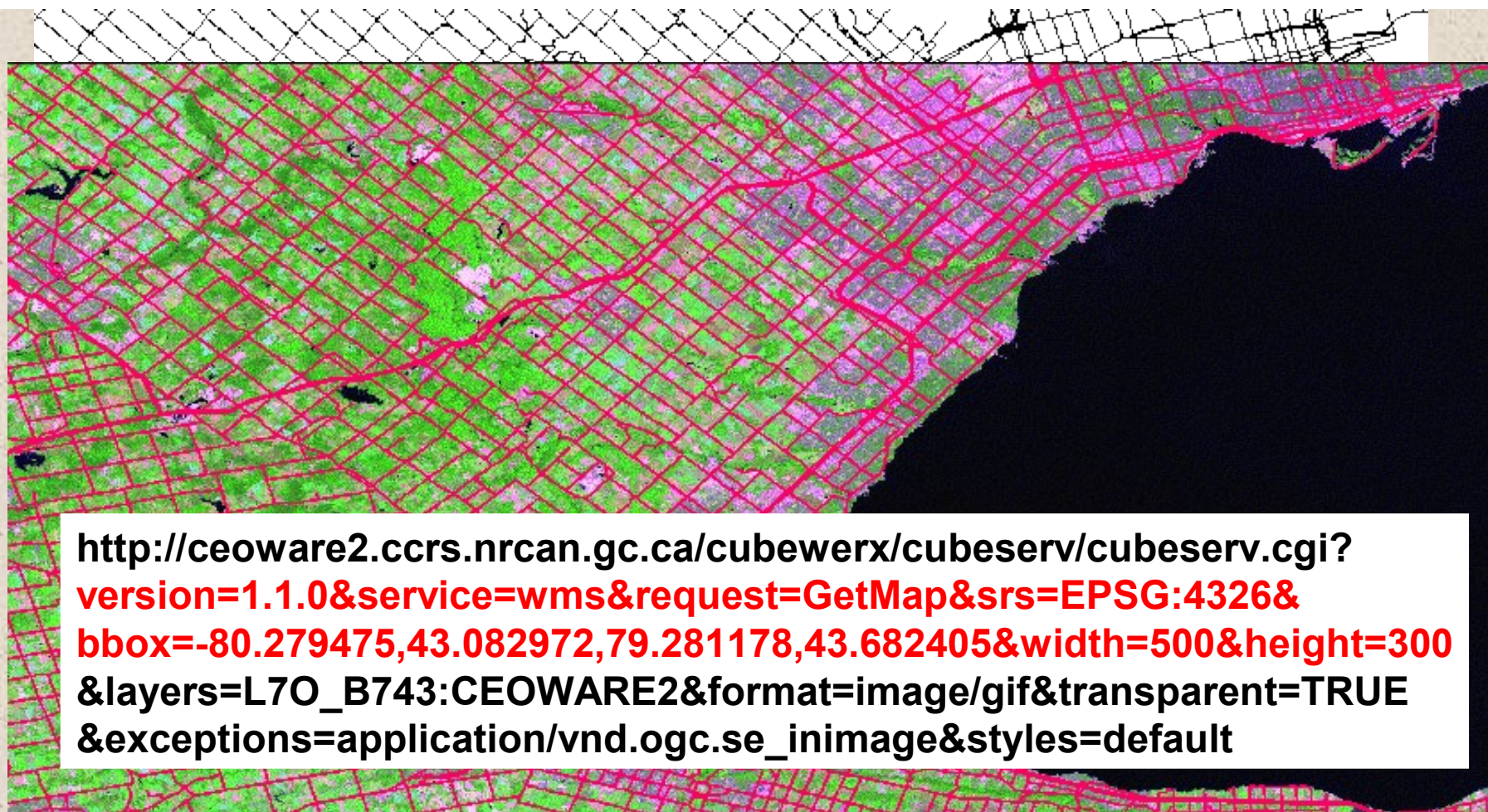
- Unified OWS Metadata!!
 - MapServer now supports “ows_*” type metadata declarations
 - OWSs with common metadata element names can use this
 - Cuts down on mapfile size
 - Less management of metadata

MapServer and OWS

- Instead of:
 - “wms_title” “Land Use”
 - “wfs_title” “Land Use”
- How about:
 - “ows_title” “Land Use”
 - Satisfies anything “[wms|wfs|wcs|gml|sos]_title”
- You can still set service-specific metadata if you need to, which overrides “ows_”

OGC WMS

- Provides images of map data defined by a geographic / spatial component
- Provides point based query functionality
- Interoperable means of map compositing from n servers
- 'just-in-time' approach



[http://ceoware2.ccrs.nrcan.gc.ca/cubewerx/cubeserv/cubeserv.cgi?
version=1.1.0&service=wms&request=GetMap&srs=EPSG:4326&
bbox=-80.279475,43.082972,79.281178,43.682405&width=500&height=300
&layers=L7O_B743:CEOWARE2&format=image/gif&transparent=TRUE
&exceptions=application/vnd.ogc.se_inimage&styles=default](http://ceoware2.ccrs.nrcan.gc.ca/cubewerx/cubeserv/cubeserv.cgi?version=1.1.0&service=wms&request=GetMap&srs=EPSG:4326&bbox=-80.279475,43.082972,79.281178,43.682405&width=500&height=300&layers=L7O_B743:CEOWARE2&format=image/gif&transparent=TRUE&exceptions=application/vnd.ogc.se_inimage&styles=default)

[http://wms.cits.rncan.gc.ca/cgi-bin/cubeserv.cgi?
version=1.1.0&service=wms&request=GetMap&srs=EPSG:4326&
bbox=-80.279475,43.082972,79.281178,43.682405&width=500&height=300
&layers=ROUTE_1:BNDT/NTDB-250K&format=image/gif&transparent=TRUE
&exceptions=application/vnd.ogc.se_inimage&styles=default](http://wms.cits.rncan.gc.ca/cgi-bin/cubeserv.cgi?version=1.1.0&service=wms&request=GetMap&srs=EPSG:4326&bbox=-80.279475,43.082972,79.281178,43.682405&width=500&height=300&layers=ROUTE_1:BNDT/NTDB-250K&format=image/gif&transparent=TRUE&exceptions=application/vnd.ogc.se_inimage&styles=default)

OGC WMS

- HTTP based (GET or POST)
- Currently version 1.1.1
- Operations
 - GetCapabilities
 - DescribeLayer
 - GetMap
 - GetFeatureInfo
 - Operation **keywords** are Case-Insensitive
 - Operation **values** are case-sensitive

OGC WMS

- GetCapabilities

- Provides XML of service functionality metadata, and layer metadata
- Parameters
 - Version (version of specification)
 - Service (multiple services may exist from this service, e.g. WMS, WFS, WCS)
 - Request (GetCapabilities)

- E.g.

- http://127.0.0.1/cgi-bin/mapserv.exe?map=/ms4w/apps/ms_ogc_workshop/service/WMS&request=GetCapabilities

OGC WMS

- Formats
 - transparency
- Exceptions
- Choose accordingly
- Layers
 - Data offerings
 - Nesting / Grouping

OGC:WMS

- DescribeLayer
 - OPTIONAL operation to provide more information about a WMS layer (WFS, etc.)
 - Parameters
 - VERSION
 - SERVICE
 - REQUEST
 - LAYERS
 - <http://127.0.0.1/cgi-bin/mapserv.exe?map=/ms4w/apps/ms4w/ogc/wms/WMS&request=DescribeLayer&layers=rivers>

Web Map Service (WMS)

- GetMap
 - Returns graphic image of data based on area of interest, data, etc.
 - Image, not features, not attributes

Web Map Service (WMS)

- Parameters
 - version (version of specification)
 - service (multiple services may exist from this service, e.g. WMS, WFS, WCS)
 - request (GetMap)
 - format (image format to be returned)
 - Get this from GetCapabilities info

Web Map Service (WMS)

- Parameters
 - bbox (spatial area of interest, i.e. minx,miny,maxx,maxy)
 - srs (reference system / projection of bounding box coordinates
 - EPSG (<http://www.epsg.org/>)
 - Get supported SRSs from GetCapabilities
 - Most widely used SRS is EPSG:4326
 - Lat/long geographic (WGS84)

Web Map Service (WMS)

- Parameters
 - width (output image width)
 - height (output image height)
 - layers (data desired to be visualized)
 - Get this from GetCapabilities info
 - Comma-separated list
 - Eg. Layers=elevation,roads,railways,...
 - First list item is bottom most output layer

Web Map Service (WMS)

- Parameters
 - TIME
 - ISO 8601 formatted timestring
 - Single point
 - Frequency
 - Range
 - Current
 - Not all WMS implementations support TIME

Web Map Service (WMS)

- Parameters
 - Styles (desired portrayal of data)
 - Get this from GetCapabilities info per layer
 - Comma-separated list
 - e.g.. layers=elevation,roads,railways&styles=default,red,blue
 - Style list MUST align with layer list
 - Empty list value for non styled layers
 - e.g. to style ONLY railways layer:
layers=elevation,roads,railways&styles=,,blue

Web Map Service (WMS)

- Parameters
 - Styles and SLD
 - SLD enables remote classification and symbolization of data
 - Overrides server-based styles if request from user
 - To use in GetMap, SLD document must exist over HTTP, and cited in GetMap request
 - &sld=http://localhost/ms_ogc_workshop/sld/rivers.sld
 - OR &sld_body=<entire_sld_document>
 - (consider HTTP POST for SLD_BODY)
 - More info on SLD later

Web Map Service (WMS)

- Parameters
 - transparent (whether to make non-opaque data pixels transparent)
 - Either TRUE or FALSE
 - Useful for layering data from multiple remote WMS services atop each other for map composition
 - Depends on image format (JPEG is not transparent)
 - Depends on web browser support
 - GIF transparency is supported in all browsers
 - PNG transparency support in newer browsers
 - bgcolor (optional background color of image)

Web Map Service (WMS)

- Parameters
 - Exceptions (how to handle errors)
 - Can be caused by:
 - Server malfunction
 - Invalid client syntax (missing required values, etc.)
 - application/vnd.ogc.se_xml
 - application/vnd.ogc.se_inimage
 - application/vnd.ogc.se_blank

Web Map Service (WMS)

- Parameters
 - Exceptions
 - Which one should I use?
 - application/vnd.ogc.se_inimage
 - Useful for easily visualizing errors in your app
 - Can also be ugly to the end-user / audience
 - application/vnd.ogc.se_blank
 - Returns a blank image
 - Difficult to recognize what type or error has occurred
 - application/vnd.ogc.se_xml
 - Returns an XML exception document
 - Difficult to decode if your app is requesting an image type
 - Choose accordingly

Web Map Service (WMS)

- E.g.:
- http://127.0.0.1/cgi-bin/mapserv.exe?map=/ms/WMS&request=GetMap&srs=EPSG:4326&bbox=-180,-90,180,90&format=png&layers=land_shallow_topo_2048,rivers&styles=,&transparent=true&width=500&height=300

Web Map Service (WMS)

- GetFeatureInfo
 - Performs point-based queries on map data
 - No ability for complex, expression-like queries
 - This is covered the WFS specification

Web Map Service (WMS)

- GetFeatureInfo
 - Parameters
 - <all GetMap parameters>
 - request=GetFeatureInfo instead of GetMap
 - Pass on ALL GetMap keyword-value pairs as if performing a GetMap request
 - x (pixel value in X image coordinates)
 - y (pixel value in Y image coordinates)
 - query_layers (layers to be queried)
 - Can be one or multiple layers
 - This does not substitute passing the layers parameter

Web Map Service (WMS)

- Parameters
 - info_format
 - Get this from GetCapabilities info
 - Common formats
 - HTML
 - » difficult to parse
 - GML.1
 - » XML-based
 - » Lacks common definition structure between vendor implementations
 - See http://127.0.0.1/ms_ogc_workshop/ for examples

OGC:WMS

- Enabling in MapServer
 - MapServer must be built with --with-wms
 - Through metadata elements in mapfile
 - “ows_*” or “wms_*” type metadata
 - These drive interface content
 - see [wms-server howto](#)

OGC:WMS

- The Server URL thing
- MapServer needs CGI “map” keyword (i.e. MS_MAPFILE environment variable) to drive interface
- This can be tacked on to server URL as the base WMS server URL prefix
- If you want, you can hide mapfile via HTTPD settings
- See:
 - http://127.0.0.1/ms_ogc_workshop/service/hide-mapfile-loc

Publishing OGC:WMS

- Go to dir: /ms4w/apps/ms_ogc_workshop/service/
- Open [exercise.map](#)
- Add all WMS contact information in mapfile WEB METADATA
- Try new server URL:
 - http://127.0.0.1/cgi-bin/mapserv.exe?map=/ms4w/apps/ms_ogc_workshop/service/exercise.map
 - Perform a GetCapabilities request

Publishing OGC:WMS

- Add the rivers layer to the mapfile to publish as a WMS layer
- See [wms-server howto](#)
- Color should 0 0 255
- Data is in
 - [/ms4w/apps/ms_ogc_workshop/data/rivers](#)

Publishing OGC:WMS

– Save to disk?

- Capabilities responses for WMS use a MIME type which web browsers are not familiar with:
 - application/vnd.ogc.wms_xml
- You can configure your web browser to launch a specific application when it encounters this MIME type
 - i.e. Firefox: add to mimeTypees.rdf:

```
<RDF:Description
  RDF:about="urn:mimetype:handler:application/vnd.ogc.wms_xml"
      NC:alwaysAsk="true"
      NC:saveToDisk="true">
  <NC:externalApplication
    RDF:resource="urn:mimetype:externalApplication:text/xml"/>
</RDF:Description>
```


OGC:SLD

- “Add-on” specification to OGC:WMS
 - “SLD-enabled WMS”
- Enables custom styling
 - Data at the server does not have to change
 - Client sends SLD XML document for symbolization, etc.
 - Either as URL or within BODY of the request
 - If URL, it MUST be resolvable and accessible by the WMS server

OGC:SLD

- Additional OGC:WMS operations with OGC:SLD:
 - GetLegendGraphic
 - GetStyles
 - PutStyles

OGC:SLD

- GetLegendGraphic
- Dynamic legend icon for a given layer
- Parameters
 - VERSION
 - SERVICE
 - REQUEST
 - FORMAT
 - LAYER
 - SLD

OGC:SLD

- GetStyles
- Returns OGC:SLD for a given layer
- Parameters
 - VERSION
 - SERVICE
 - REQUEST
 - LAYERS

OGC:SLD

- PutStyles
- Stores SLD document on WMS server
- Not supported by MapServer

OGC:SLD

- Enabling in MapServer
- Not much, really
 - MapServer code (CGI) basically exposes and converts CLASS objects to SLD constructs
 - Also via PHP MapScript to mapObj or layerObj
 - See here for SLD examples:
 - http://127.0.0.1/ms_ogc_workshop/index.html

Consuming OGC:WMS

- MapServer, on your behalf, can connect to OGC:WMS as well
- Takes care of client code specifics (whew!)
- Specific LAYER METADATA elements in mapfile
- See [wms-client-howto](#)
- http://127.0.0.1/ms_ogc_workshop/client/wms/demo_int
- http://127.0.0.1/ms_ogc_workshop/client/wms/demo.ma

Consuming OGC:WMS

- Go to dir:
 - [/ms4w/apps/ms_ogc_workshop/client/wms/satellite/](#)
- Add a WMS layer to demo.map
 - World Soil Resources
 - Server:
 - » <http://geonetwork3.fao.org/cgi-bin/wms?map=/GeoNetwork>
 - Layer name:
 - » world_soil_resources
 - Hints:
 - Do a GetCapabilities to get more info on the layer, formats, projections, version, etc.
 - See [wms-client-howto](#)
 - More examples of remote global layers in:
 - » http://127.0.0.1/ms_ogc_workshop/client/wms/satellite/servers.txt

Consuming OGC:WMS and SLD

- Go to dir:
 - /ms4w/apps/ms_ogc_workshop/client/wms/satellite/
- Add another WMS layer to demo.map
 - Bird Studies Canada Important Bird Locations
 - Server:
 - » http://www.bsc-eoc.org/cgi-bin/bsc_ows.asp
 - Layer name:
 - » IBA
- Add the following SLD document:
 - » http://devgeo.cciw.ca/ms_tmp/iba_sld.xml
- Hints:
 - Look at the SLD document to get a feel of what's going on
 - See [sld-howto](#) and [workshop SLD examples](#)
 - If you're ambitious, set "wms_sld_body" to "AUTO", and generate a CLASS object, which generates the SLD on the fly in the WMS GetMap request

OGC:WFS

- Feature level access to spatial data (vectors)
- Rich query interface
- Returns GML
- Transactional capability
- Security considerations for OGC:WFS-T
 - MapServer supports basic WFS
 - No transactions

OGC:WFS

- Operations
 - GetCapabilities
 - DescribeFeatureType
 - GetFeature

OGC:WFS

- GetCapabilities
- Same idea as OGC:WMS GetCapabilities
- Parameters
 - VERSION
 - SERVICE
 - REQUEST
- <http://127.0.0.1/cgi-bin/mapserv.exe?map=/ms4w/apWFS&request=GetCapabilities>

OGC:WFS

- DescribeFeatureType
- Provides an outline of the structure of a feature type (fields, etc.)
- Analogous to SQL describe <table> command
- Parameters
 - VERSION
 - SERVICE
 - REQUEST
 - TYPENAME
- http://127.0.0.1/cgi-bin/mapserv.exe?map=/ms4w/apps/ms_o_WFS&request=DescribeFeatureType&typename=rivers

OGC:WFS

- GetFeature
- Gimme data!
- Parameters
 - VERSION
 - SERVICE
 - REQUEST
 - TYPENAME
 - FILTER (optional)
 - BBOX (can also be done through FILTER)
 - E.g.: see [index.html](#)

OGC:WFS

- Enabling in MapServer
 - MapServer must be built with --with-wfs
 - Through metadata elements in mapfile
 - “ows_*” or “wfs_*” type metadata
 - These drive interface content
 - see [wfs-server howto](#)
 - Layers must contain “DUMP TRUE”

Consuming OGC:WFS

- MapServer, on your behalf, can connect to OGC:WFS as well
- Takes care of client code specifics (whew!)
- Specific LAYER METADATA elements in mapfile
- See [wfs-client-howto](#)
- http://127.0.0.1/ms_ogc_workshop/client/wfs/demo_init
- http://127.0.0.1/ms_ogc_workshop/client/wfs/demo.map

OGC:GML

- Text-based, portable data format
- Self-describing, XML
- GML schemas define geospatial objects for you
- YOU define what's specific to your data
 - attributes, etc.

OGC:GML

- Any GML parser can read your data!!
- Smarter components which are based on the GML standard

Serving OGC:GML

- GML usually comes from a OGC:WFS
- MapServer can also serve up static files
 - OGR supports GML2
 - Certain structure required for OGR to recognize content model
 - Dataset/record type hierarchy
- You can serve up GML:
 - From GML documents you made yourself
 - From GML documents you copied from elsewhere
 - By doing nothing; OGC:WFS offers GML already

Serving OGC:GML

- Making it happen
 - CONNECTIONTYPE OGR
 - CONNECTION “/path/to/gml/document”
 - The rest is like any other LAYER object definition in the mapfile
- OGR processing
 - Creates and caches *.gfs overview type file when first reading the data
 - Quicker for subsequent data scans

Serving OGC:GML

- MapServer now supports GML3!
 - via WFS supports both GML2 and GML3
 - Same as serving GML2 for WFS
 - WFS requests should have:
 - » DescribeFeatureType
“outputFormat=XMLSCHEMA”
 - » DescribeFeatureType
“outputFormat=SFE_XMLSCHEMA”
 - » GetFeature “outputFormat=GML3”
 - » GetFeature “outputFormat=GML2”

Serving OGC:GML

- Check out [mums.gml](#) for an example
- Check out [mums.xsd](#) for the XML Schema definition
 - MapServer doesn't necessarily use the .xsd
 - Good practice nonetheless for validating XML parsers

Serving OGC:GML

- Try creating your own document by copying/pasting the mums examples
- Update reference to your new data in [demo.map](#)
- Now you're a GML guru 😊

OGC:Filter

- “Add on” specification to OGC:WFS
- Custom XML query language
- SQL in XML, almost
- Spatial and aspatial query capabilities
 - Logical
 - Spatial
 - Comparative

OGC:Filter

- SQL:
 - select * from roads where roadtype = 1
- OGC:Filter:

```
<Filter>  
  <PropertyIsEqualTo>  
    <PropertyName>roadtype</PropertyName>  
    <Literal>1</Literal>  
  </PropertyIsEqualTo>  
</Filter>
```

OGC:Filter

- Enabling in MapServer
- Not much, really
 - Handled by MapServer OGC:WFS code
 - E.g.:
 - http://127.0.0.1/ms_ogc_workshop/filter/index.html
 - See [MapServer Filter howto](#)

Exercise

- Add WFS client layer and client side filter encoding to /ms4w/apps/ms_ogc_workshop/client/wfs-filter/demo.map
- Use OWS server:
 - http://www2.dmsolutions.ca/cgi-bin/mswfs_filter
 - Add as remote WFS layer
 - Create the following filters:
 - WHERE NAME = 'Digby'
 - WHERE NAME LIKE 'Syd%'
 - WHERE NAME = 'Digby' or NAME LIKE 'Syd%'
- Application
http://127.0.0.1/ms_ogc_workshop/client/wfs-filter/demo_init.html
- Helpers
 - http://www2.dmsolutions.ca/cgi-bin/mswfs_filter?SERVICE=WFS&version=1.0.0&Request=GetCapabilities
 - http://www2.dmsolutions.ca/cgi-bin/mswfs_filter?SERVICE=WFS&version=1.0.0&Request=DescribeFeatureType&TYPENAME=popplace
 - See [Filter examples](#) for help
 - See [MapServer Filter howto](#) and [MapServer WFS client howto](#)

result map map file /ms4w/apps/ms_ogc_workshop/client/wfs-filter/demo_answer.map

result application : http://127.0.0.1/ms_ogc_workshop/client/wfs-filter/demo_init_answer.html

LAYER

NAME popplace

GROUP "VECTOR"

TYPE POINT

STATUS ON

CONNECTIONTYPE WFS

CONNECTION "http://www2.dmsolutions.ca/cgi-bin/mswfs_filter?"

PROJECTION

"init=epsg:42304"

END

METADATA

"wfs_connectiontimeout" "60"

"wfs_version" "1.0.0"

"wfs_service" "WFS"

"wfs_typedname" "popplace"

"wfs_latlonboundingbox" "-65.4238 43.3796 -60.2557 47.7669"

WHERE NAME = 'Digby'

#"wfs_filter" "<PropertyIsEqualTo><PropertyName>NAME</PropertyName><Literal>Digby</Literal></PropertyIsEqualTo>"

LABELITEM "NAME"

CLASS

NAME "popplace"

STYLE

COLOR 255 0 0

SYMBOL 'circle'

SIZE 6

END

LABEL

COLOR 255 0 0

FONT fritqat-italic

TYPE truetype

SIZE 8

POSITION AUTO

PARTIALS FALSE

END

END #class

END#layer

OGC:WCS

- The raster equivalent to OGC:WFS
- Provides “real” raster data
 - DEM
 - GeoTIFF

OGC:WCS

- Operations
 - GetCapabilities
 - DescribeCoverage
 - GetCoverage

OGC:WCS

- GetCapabilities
- Same idea as OGC:WMS GetCapabilities
- Parameters
 - VERSION
 - SERVICE
 - REQUEST
- <http://127.0.0.1/cgi-bin/mapserv.exe?map=/ms4w/apWCS&request=GetCapabilities>

OGC:WCS

- DescribeCoverage
- Provides an outline of the structure of a coverage
 - bands
 - resolution
- Parameters
 - VERSION
 - SERVICE
 - REQUEST
 - COVERAGE
- http://127.0.0.1/cgi-bin/mapserv.exe?map=/ms4w/apps/ms_o WCS&request=DescribeCoverage&coverage=toronto

OGC:WCS

- GetCoverage
- Gimme data!
- Parameters
 - VERSION
 - SERVICE
 - REQUEST
 - COVERAGE
 - CRS
 - BBOX
- E.g.: see [index.html](#)

OGC:WCS

- Enabling in MapServer
 - MapServer must be built with --with-wcs
 - Through metadata elements in mapfile
 - “ows_*” or “wcs_*” type metadata
 - These drive interface content
 - Layers must contain “DUMP TRUE”
 - see [wcs-server howto](#)

OGC:SOS

- Provides Observations and Measurements
 - Sensors
 - In-situ
 - Monitoring
 - e.g. water quality / quantity, weather, air quality
- Can Support / Enable:
 - Data extraction
 - Graphing and trend analysis

OGC:SOS

- Operations
 - GetCapabilities
 - DescribeSensor
 - GetObservation

OGC:SOS

- GetCapabilities
- Same idea as OGC:WMS GetCapabilities
- Parameters
 - VERSION
 - SERVICE
 - REQUEST
- <http://127.0.0.1/cgi-bin/mapserv.exe?map=/ms4w/apSOS&request=GetCapabilities>

OGC:SOS

- DescribeSensor
- Provides an outline of the structure of a coverage
 - bands
 - resolution
- Parameters
 - VERSION
 - SERVICE
 - REQUEST
 - SENSORID
- http://127.0.0.1/cgi-bin/mapserv.exe?map=/ms4w/apps/ms_oSOS&request=DescribeSensor&procedure=SOMEID

OGC:SOS

- GetObservation
- Gimme data!
- Parameters
 - VERSION
 - SERVICE
 - REQUEST
 - SENSORID
 - CRS
 - BBOX
 - E.g.: see [index.html](#)

OGC:SOS

- Enabling in MapServer
 - MapServer must be built with --with-sos (and prerequisites [libxml2])
 - Through metadata elements in mapfile
 - “ows_*” or “sos_*” type metadata
 - These drive interface content
 - Layers must contain “DUMP TRUE”
 - see [sos-server howto](#)

OGC:WMC

- Web Map Context Documents
- Saves Web Mapping Application State
 - remote WMS layer pointers
- Like “project” files in common desktop GIS
- XML-based

OGC:WMC

- Enabling in MapServer
- Check out [mapcontext howto](#)
 - Handled by MapServer CGI code
 - ‘hidden’ API param called “request=GetContext”
 - turned OFF by default
 - PHP Mapscript methods
 - saveMapContext
 - loadMapContext
- E.g. and mapfile with context parameters defined:
 - http://127.0.0.1/ms_ogc_workshop/context/demo.map
 - http://127.0.0.1/ms_ogc_workshop/index.html

OGC:WMC

- Enabling in MapServer
- Now MapServer CGI can ingest context
- CGI param “context=<url>”
- Check out [mapcontext howto](#)

Mapscript WxS

- Allows for custom scripting of OGC Web Services via mapscript (perl, python, php, etc.)
- Can add intermediary layer of logic between OGC requests and responses
 - access control
 - custom metadata
- See [MapScript WxS howto](#)

Wxs : PHP example

- http://localhost/ms_ogc_workshop/mapscript/php/ows.php?service=WMS&version=1.1.1&Request=GetCapabilities

```
<?php
dl("php_mapscript.dll");
$request = ms_newowsrequestobj();
$request->loadparams();
/*forcing the version from 1.1.1 to 1.1.0 */
$request->setParameter("VeRsIoN","1.1.0");
ms_iostreamstdoutbuffer();
$oMap = ms_newMapobj("../..service/exercise_answer.map");
$oMap->owsdispatch($request);
$contenttype = ms_iostreamstdoutbuffercontenttype();
$buffer = ms_iogetstdoutbufferstring();
header('Content-type: application/xml');
echo $buffer;
ms_ioresethandlers();
?>
```

Consuming Remote Web Services

- Distributed
 - Data stays at the source
 - You're getting a 'view' of user-specified chunk of data
- “middleware”
 - Connect to Web Services with software (Perl, Java, PHP, etc.)
 - Consume XML
 - Reformat into HTML
 - Use JavaScript for interactive navigation
- E.g. Connecting to a world gazetteer WFS to 'zoom to placename':
 - http://127.0.0.1/ms_ogc_workshop/client/webservices/demo_init.html

Consuming Remote Web Services

- Try rolling your own!
 - Add calling button in [demo.html](#)
 - Create middleware
 - Copy/paste [gazetteer.php](#) to start off
 - Choose a web service to connect to
 - Format request and input params
 - Figure out XML output structure
 - Output with HTML
 - Adjust JavaScript if needed
- Examples of XML Web Services and sample requests at:
 - [webservices.txt](#)

Nice-to-Haves

- Official OGC Conformance
 - Rubber stamping
- Legends for Raster layers
- OGC:WMC
 - Allowing exposure of local data in a context document
- OGC:Filter
 - Enhanced support for spatial ops
 - Can we use GEOS?

Future Goodies

- Next Generation OGC Services:
 - OWS Common
 - Unified Constructs
 - Metadata
 - Operations
 - Exceptions
 - Now adopted (1.0.0, 1.1.0)
 - Will be implemented in OGC specs
 - WCS 1.1, WFS 1.1, WMS 1.3
- OGC SensorWeb is coming
 - Thoughts on SensorML, O & M, SOS

Issues

- “Where can I find OGC Web Services”
- Canada
 - <http://geodiscover.cgdi.ca/gdp/search?action=searchForm&entryType=webService>
- USA
 - <http://edcw2ks15.cr.usgs.gov/fgdc/EDCgateway.html>
 - <http://gcmd.nasa.gov/>
- More
 - http://www.skylab-mobilesystems.com/en/wms_serverlist.html
 - <http://wms-sites.com/>
- User-friendliness of discovery of OGC layers/features/services, etc.

Contributors

- **Yewondwossen Assefa**
 - yassefa at dmsolutions.ca
- **Jean Francois Doyon**
 - jdoyon at nrcan.gc.ca
- **Tom Kralidis**
 - tom.kralidis at ec.gc.ca
- **Steve Lime**
 - steve.lime at dnr.state.mn.us
- **Daniel Morissette**
 - dmorissette at mapgears.ca
- **Jeff McKenna**
 - jmckenna at gatewaygeomatics.com
- **Bart van den Eijnden**
 - bartvde at osgis.nl

Thanks and Good Luck!

<http://ms-ogc-workshop.maptools.org/>