Using python client libraries to access remote servers via Web Coverage Services

Ben Domenico, Unidata, UCAR
Dominic Lowe, British Atmospheric Data Center

and
The GALEON Team

January 2009

Abstract

As part of the GALEON 2 (Geo-interface for Air, Land, Environment, Ocean NetCDF) interoperability experiment of the OGC (Open Geospatial Consortium), a Web Coverage Service (WCS) client library has been written in the python language. The WCS client library, developed at the British Atmospheric Data Centre (BADC) is available as part of the OWSlib python package, an OGC Web Service utility library for working with OGC map, feature, and coverage services. OWSlib provides a common API for accessing service metadata (capabilities) from WMS/WCS/WFS servers and wrappers for GetMap, GetFeature, and GetCoverage requests. A script for exercising rudimentary data access was created for use with GALEON servers at a number of sites, including the US Pacific Fisheries Environmental Laboratory, ARGOSS of the Netherlands, the USGS at Woods Hole, UCAR Unidata, NCDC NOMADS, the KNMI of the Netherlands, the US NSIDC, the Italian CNR-IMAA, the UK NERC, the US NNEW Project, and George Mason University. The servers include implementations based on several release levels: WCS 1.0.0, 1.1.0, and 1.1.1. The paper reports on lessons learned regarding the use of python scripts for exercising standards-based protocols, issues encountered relating to the different release levels of the WCS protocol, as well as problems uncovered and corrected in the python libraries.

GALEON Background

The OGC (Open Geospatial Consortium) GALEON IE (Geo-interface for Air, Land, Earth, Oceans NetCDF Interoperability Experiment) has expanded its objectives considerably beyond its initial, focused set of goals. However, the overall mission remains the same: specify and use standard interfaces to foster interoperability between data systems used by the traditional GIS community and those in the community we refer to as the Fluid Earth Sciences (FES, mainly oceanography and atmospheric science). This strategic target is becoming increasingly important as FES observations and forecasts are achieving the high spatial resolutions (a few kilometers and better) of the GIS realm. The challenge is to enable the practitioners in each realm to continue using the powerful tools available through their traditial "stovepipe" applications while allowing for integration of data and applications between the two realms by developing and employing standard, web services-based interfaces between the two.

GIS Realm

In general (and somewhat oversimplified) terms, the GIS community thinks of datasets as collections of static *features* -- e.g., roads, lakes, plots of land -- with geographic footprints on the Earth's surface. The *features* are discrete objects with attributes which can be stored and manipulated conveniently in a **database**. Examples of data types in Earth sciences related GIS data collections include topography, bathymetry, hydrological data (streamflow and water quality), output of flood and landslide models, land use and surface characteristics, soil moisture, demographics, infrastructure.

A typical example of a GIS-rendered map is shown below. It's easy to see how the items in the legend can be stored as tables in a relational database system and rendered as "layers" on a map.

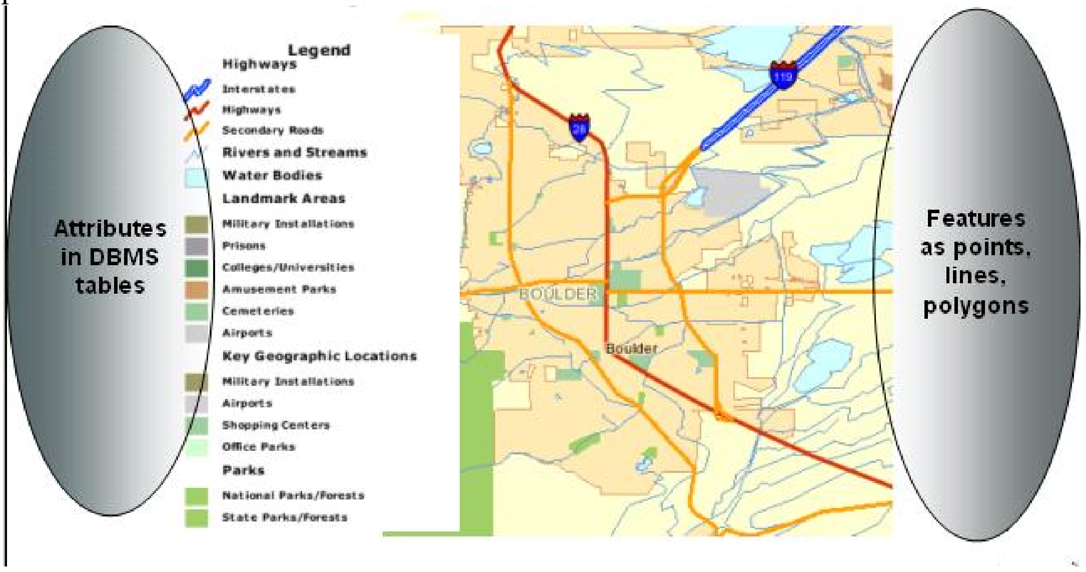


Figure 1: Typical GIS rendering of "features" projected onto map surface of the Earth (Image courtesy Tiger 2000 Map Service)

Fluid Earth Sciences (FES) Realm

To the FES community, the world is characterized by a set of *parameters* (e.g., pressure, temperature, wind speed) which vary as continuous functions in 3-dimensional space and time. The behavior of the *parameters* in space and time is governed by the laws of fluid dynamics as represented by a set of **partial differential equations.** In this world view, data are simply discrete points in the mathematical function space. Examples of common data types in Earth sciences data collections related to the fluid Earth are:

- volume scans from ground-based radars
- satellite imagery
- output of numerical forecast models (weather, climate, ocean circulation, storm surge)

- weather station observations
- vertical soundings (balloon, aircraft, ocean depth probes, etc.)
- trajectories such as those generated by observing systems onboard aircraft.

An animated visualization of the output of a numerical weather forecast model is show in the following figure. It illustrates the true multidimensionality of the dataset with the jet stream rendered as a "solid" time-varying object in 3 spatial dimensions. In addition, other variables such as temperature (at a given level) and pressure are shown as a surface and line contours in both vertical and horizontal cross-sections. The map background in this particular image was generated from GIS shapefiles.

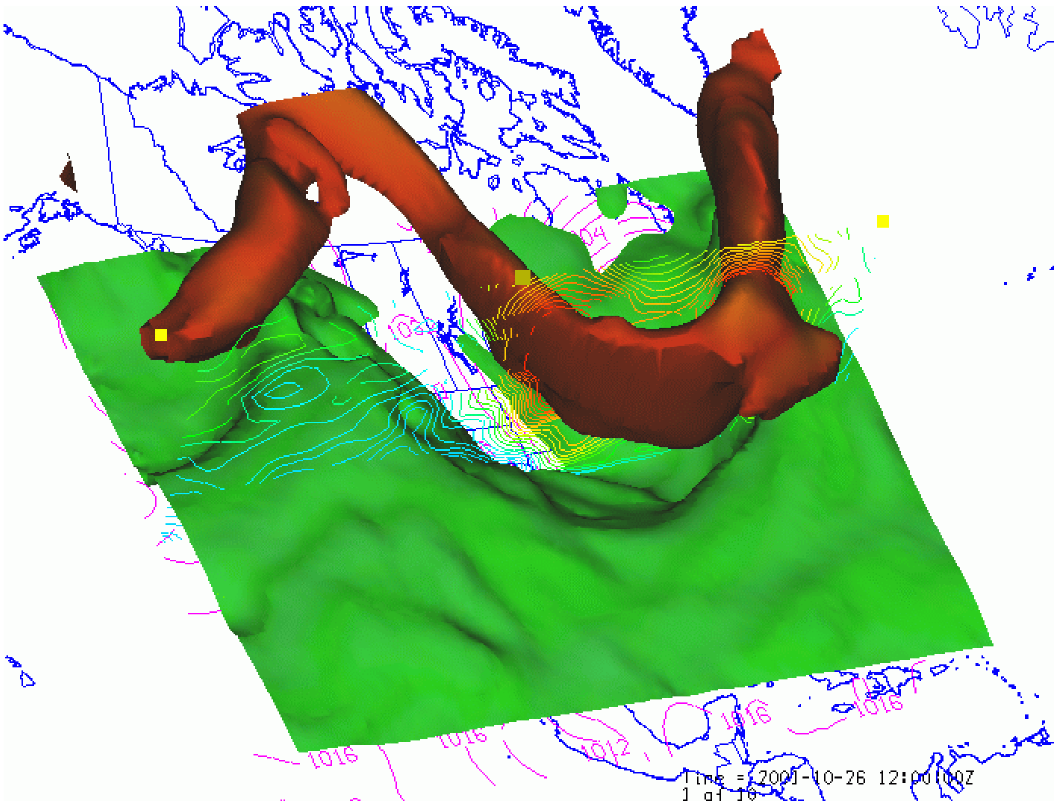


Figure 2: FES visualization of weather forecast model output (Generated by Unidata Integrated Data Viewer)

Powerful Tools in Both Realms

Integrated visualization is one of the first things that comes to mind as a reason for making the data systems interoperable. Furthermore, it is important to be able to view the disparate datasets from within the same package. Indeed much progress is already being made in terms of being able to create integrated overlays of datasets from the two realms. However, there are other crucial capabilities in GIS and FIS tools that are just as important. Because of the underlying data model that facilitates the solutions to the equations of fluid dynamics, numerical forecasting tools are a strength in the FES realm. So predicting the path of severe storms is routine. On the other hand, the underlying databases in GIS facilitate responses to queries relating to overlap among various types of georeferenced features. Thus, if the predicted path of a storm can be characterized in GIS terms, GIS tools can readily determine how many people live in the area and what infrastructure is likely to be affected. That's not to say that the need is for a flow of data only in the one direction.

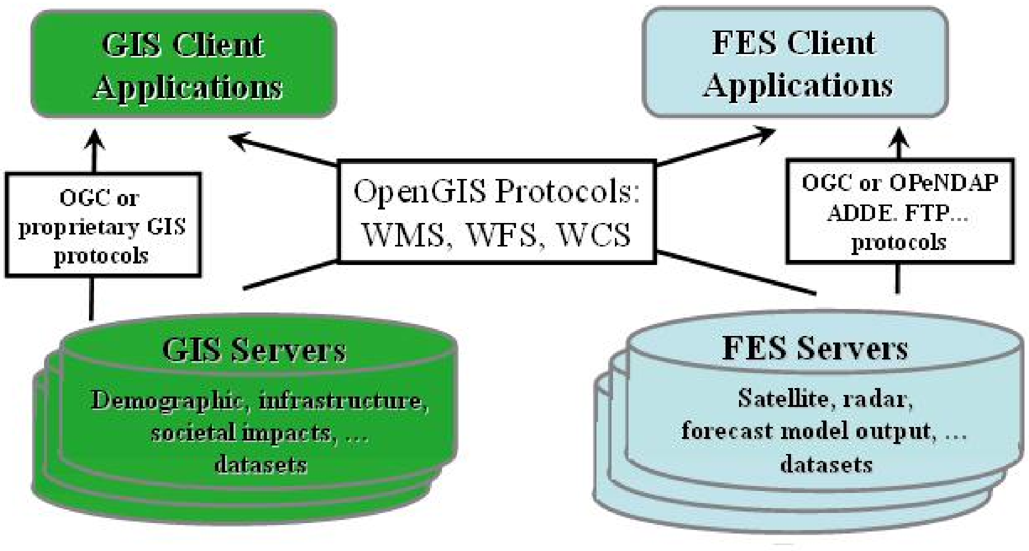
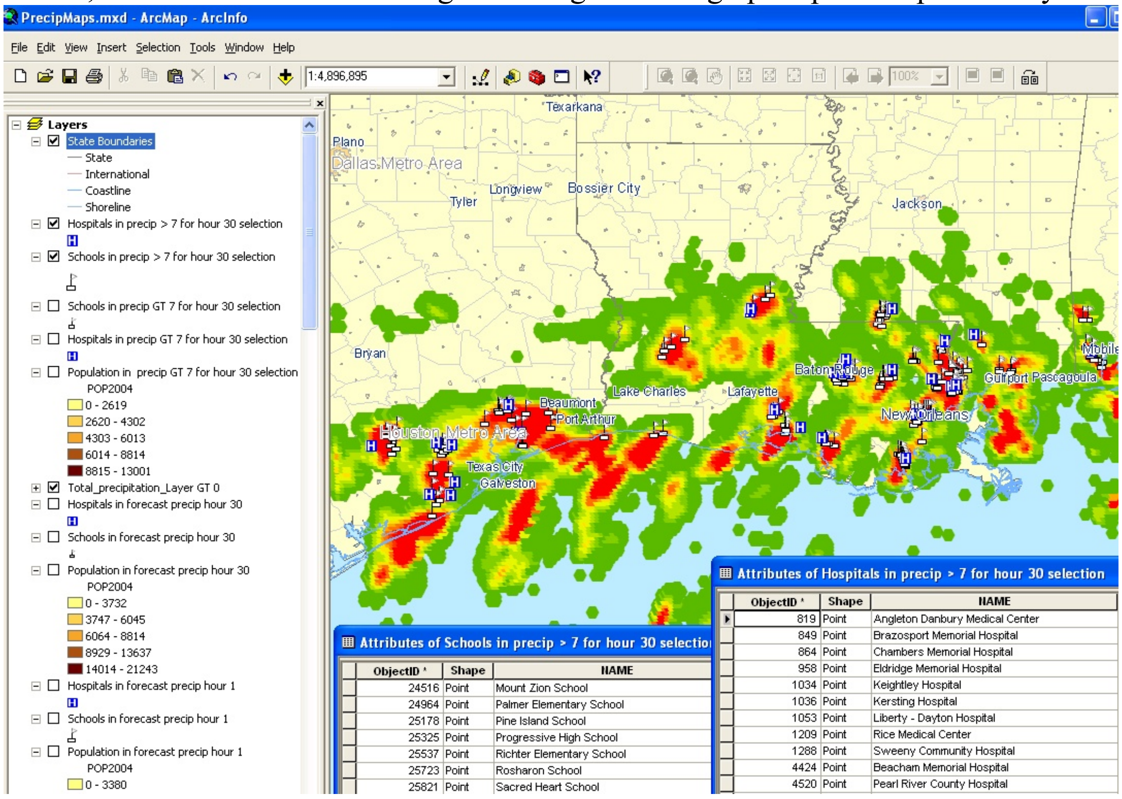

GIS Client Applications
FES Client Applications
OGC or proprietary GIS protocols
OpenGIS Protocols: WMS, WFS, WCS
OGC or OPeNDAP ADDE, FTP... protocols
GIS Servers
Demographic, infrastructure, societal impacts, ... datasets
FES Servers
Satellite, radar, forecast model output, ... datasets

Figure 3: Schematic of GIS-FES inteoperability via standard interfaces

Integrated Processing and Visualization Tools

To gain a sense of the power afforded by integrating the tools of both communities, the screen shot below shows demographic information as an overlay on real-time, high resolution, local forecast models being run in regions of high precipitation probability.


PrecipMaps.mxd - ArcMap - ArcInfo
File Edit View Insert Selection Tools Window Help
1:4,896,895
Layers
State Boundaries
State
International
Coastline
Shoreline
Hospitals in precip > 7 for hour 30 selection
Schools in precip > 7 for hour 30 selection
Schools in precip GT 7 for hour 30 selection
Hospitals in precip GT 7 for hour 30 selection
Population in precip GT 7 for hour 30 selection
POP2004
0 - 2619
2620 - 4302
4303 - 6013
6014 - 8814
8815 - 13001
Total_precipitation_Layer GT 0
Hospitals in forecast precip hour 30
Schools in forecast precip hour 30
Population in forecast precip hour 30
POP2004
0 - 3732
3747 - 6045
6064 - 8814
8929 - 13637
14014 - 21243
Hospitals in forecast precip hour 1
Schools in forecast precip hour 1
Population in forecast precip hour 1
POP2004
0 - 3380
Texarkana
Plano
Dallas Metro Area
Tyler
Longview
Bossier City
Jackson
Bryan
Lake Charles
Lafayette
Baton Rouge
Gulfport Pascagoula
Mobile
New Orleans
Beaumont
Port Arthur
Houston Metro Area
Texas City
Galveston

Attributes of Schools in precip > 7 for hour 30 selection

ObjectID *	Shape	NAME
24516	Point	Mount Zion School
24964	Point	Palmer Elementary School
25178	Point	Pine Island School
25325	Point	Progressive High School
25537	Point	Richter Elementary School
25723	Point	Rosharon School
25821	Point	Sacred Heart School

Attributes of Hospitals in precip > 7 for hour 30 selection

ObjectID *	Shape	NAME
819	Point	Angleton Danbury Medical Center
849	Point	Brazosport Memorial Hospital
864	Point	Chambers Memorial Hospital
958	Point	Eldridge Memorial Hospital
1034	Point	Keightley Hospital
1036	Point	Kersting Hospital
1053	Point	Liberty - Dayton Hospital
1209	Point	Rice Medical Center
1288	Point	Sweeny Community Hospital
4424	Point	Beacham Memorial Hospital
4520	Point	Pearl River County Hospital

Figure 4: ArcMap rendering schools and hospitals region of high precipitation forecast

The forecast was generated as part of the LEAD (Linked Environments for Atmospheric Discovery) Large ITR project by the WRF (Weather Research Forecast) model on a LEAD node at Unidata. ArcMap tools were used to determine which schools and hospitals were in the areas where the high resolution WRF forecast storm total precipitation in excess of 7 inches. This illustrates the power of combining real-time (hourly) weather forecast data with the database tools of the GIS community. Similar overlays could be done with GIS data representing watershed and drainage basins to combine the atmospheric predictions with hydrological measurements.

Standards-based Access to Existing Data Systems

The overall GALEON target is this general goal of interoperability via standards-based web services interfaces. But there is one rather more specific objective that involves using these interfaces as a means for enabling traditional GIS applications to access the variety of datasets available in existing servers in the FES community. These servers which number in the hundreds are based on a set of client-server protocols that have evolved in the FES community over the last decade. The basic building blocks are NetCDF, OPeNDAP, ADDE, and THREDDS technologies. But there are other services built on these, for example LAS, GDS, and INGRID. There are already several hundred of these servers making a wide-variety and large volume of data available to existing client applications. So a key aim of GALEON is to expand the usefulness of these servers by adding a standards-based interface to provide a gateway so that WCS clients can access the datasets.
The diagram below is a schematic of this gateway implementation as it was envisioned at the time GALEON was initiated. Since that time, development work has integrated the underlying THREDDS/OPenDAP services into a package called the THREDDS Data Server (TDS) which has a rudimentary WCS interface built in.

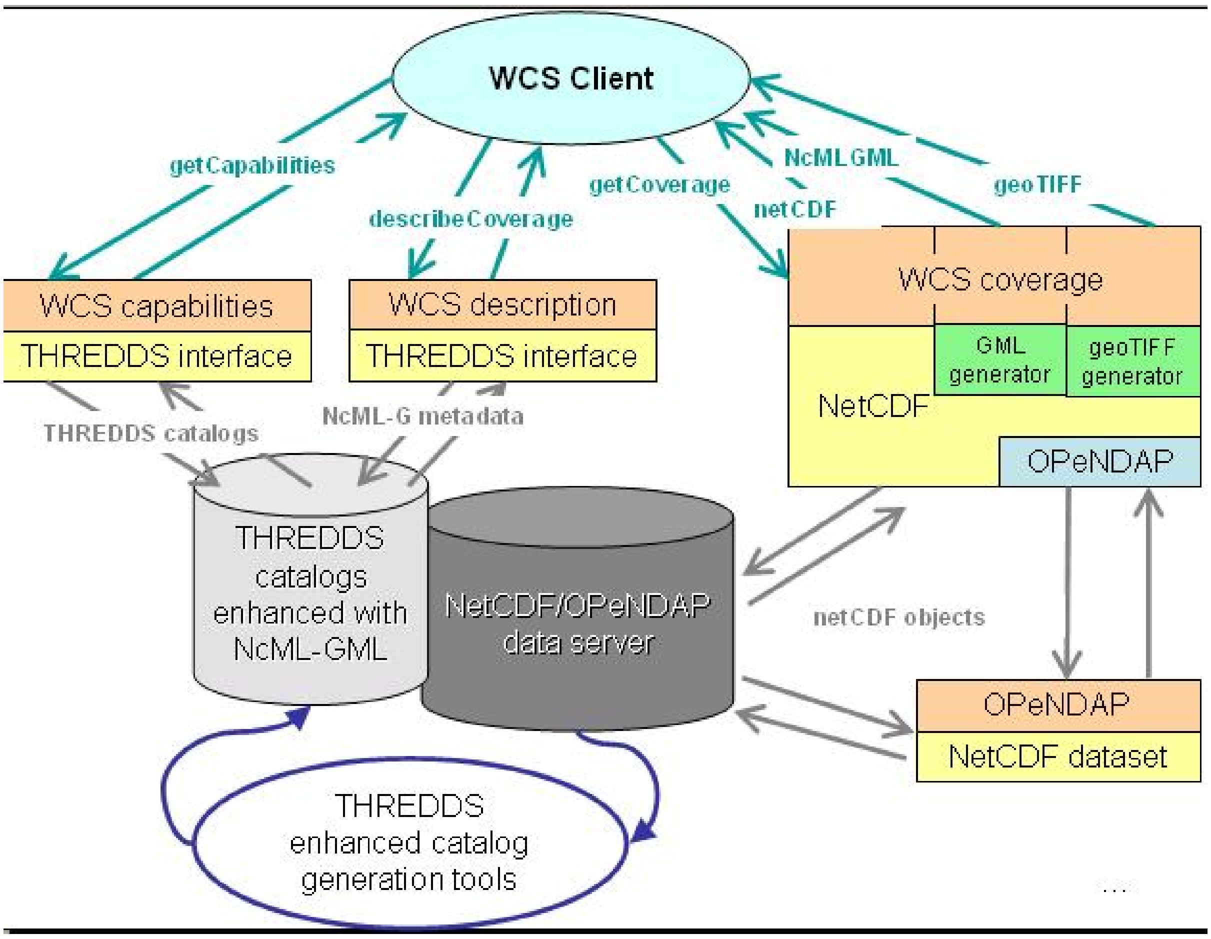


Figure 5: Initial concept of WCS-interface as a gateway to existing FES services

It is important to recognize that the gateway described above is not the sole objective of GALEON by any means. In fact, many of the most interesting and potentially most productive ideas have come up since GALEON was formally proposed. Some of these found their way into the initial use cases, some have been undertaken as the IE progressed, and others were formulated based on early experiences.

GALEON IE Evolution

At the outset, the goal of GALEON was to implement a geo-interface to netCDF datasets via the WCS 1.0 protocol specification. The WCS was to be a layer above a set of client/server and catalog protocols already widely in use in the atmospheric and oceanographic sciences communities. In particular, it leveraged the widespread base of OPeNDAP servers that provide access to netCDF datasets and accompanying THREDDS servers providing ancillary information about the datasets. The experimentation of GALEON Phase 1 established the feasibility of adapting data and metadata originating from OPeNDAP/THREDDS servers to the WCS specifications with CF-netCDF as an encoding format.
The initial experiments focused on collections of numerical forecast model output which consist of what are sometime referred to as five dimensional or 5D grids with multiple parameters (e.g., temperature, pressure, relative humidity) varying in three spatial dimensions

with two time coordinates (model run time and forecast time). Thus GALEON 1 is an early step in the direction of interoperability with data systems already in existence in the oceanographic and atmospheric sciences.

An outline of the integration path is given in:
http://my.unidata.ucar.edu/content/projects/THREDDS/OGC/WCS-THREDDS%20Gateway.htm.
Most of the activity of the first phase of GALEON focused on testing the interactions between WCS clients and servers for netCDF datasets, modifying those client and server implementations based on the testing, and recommending modifications and augmentations of the relevant OGC interfaces where appropriate. The many experiments between independently-developed client and server implementations showed that the WCS protocol does indeed have the basic functionality needed to enable access to traditional netCDF datasets. Pointers to individual GALEON experiments are given in the full **GALEON Phase 1 Summary Report** (http://www.unidata.ucar.edu/projects/THREDDS/GALEON/Reports/GALEONphase1Report.htm)
Active participants in phase 1 included the University of Florence and IMAA-CNR, the International University Bremen (now Jacobs University), George Mason University, CadCorp, RSI UK (now ITTVis), **NERC Natural Environment Research Council/ British Atmospheric Data Center,** Unidata/UCAR,, Interactive Instruments, the **NASA Geospatial Interoperability Office** Washington University St. Louis**, the U.S. NOAA Pacific Fisheries Environmental Laboratory, and NCDC, the National Climatic Data Center.**

Recommended Modifications to OGC Interface Specifications

Based on GALEON Phase1, several changes were suggested for the WCS protocol specification:

- Multiple "variables" or "parameters" in a coverage (e.g., pressure, temperature, etc.)
- Coverages with 3 spatial dimensions
- Coverages with multiple time dimensions (e.g. forecast time in model output)
- Non-spatial "height" dimension, (e.g., atmospheric pressure, ocean density)
- Irregularly-spaced grids

Concise GALEON Update

GALEON Phase 1 resulted in many successful experiments with WCS 1.0, but it was constrained exclusively to gridded data -- mainly the output of numercal forecast models, but also some satellite data. Phase 2 has focused on WCS 1.1 and later versions with some experimentation using CS-W (Catalog Services for the Web) to determine the role of catalogs vs. that of the WCS getCapabilities request. Phase 2 also ventured into discussions of whether WCS is an appropriate protocol for serving non-gridded data collections of netCDF observational data or whether that is done better via other Services, e.g., WFS and SOS.

But, one of the main issues encountered in Phase 2 has been the dearth of client implementations for WCS 1.1. This has slowed progress in conducting a set of true interoperability experiments along the lines of what was done in Phase 1. The development of a WCS component for the OWSLib appears to be at least a partial solution to this problem in that this Python library makes it possible to create simple WCS clients in the form of Python scripts. These can be tailored to exercise different aspects of the WCS protocol. The remainder of this article focuses on the OWSLib -- in particular its WCS facet -- and a few rudimentary exercises conducted with GALEON WCS servers.

Background Information on OWSLib - OGC Web Service utility library

OWSLib is a lightweight package for working with OGC map, feature, and coverage services using Python. It provides a common API for accessing service metadata and wrappers for GetCapabilities, GetMap, GetCoverage and GetFeature requests. OWSlib has been designed to:

- instantiate an OGC web service proxy
- make a GetCapabilities request and marshal the response into a Python structure that models OWS Common metadata
- proxy GetMap/GetFeature/GetCoverage/Get* requests to the service with Pythonic wrappers for those operations

OWSLib was originally developed by Sean Gillies as part of the Python Cartographic Library. Since then, it has evolved in the following versions:

- Version 0.2 contained support for accessing WMS version 1.1.1 and WFS 1.0.0 servers.
- Version 0.3 developed by Sean Gillies & Dominic Lowe introduced support for WCS versions 1.0.0 and 1.1.0.
- Version 0.3 also introduced a harmonized OWS Common-like API to service metadata that is common across WCS, WMS and WFS. This API is a 'pythonic' interpretation of OWS-Common which provides easy access to the OWS-common metadata.

Faced with the typical software library design decisions, the OWSLib team decided that key goals would be to make OWSLib lightweight and easy to use. The API has been modelled on the OGC OWS-common but not verbatim. When faced with tensions between the OGC and Python approaches, the guiding principle is that "pythonic beats pedantic as long as significant meaning is not lost" which reinforces the ease-of-use objective. The library makes use of python idioms, e.g. convenient dictionary like access to coverage metadata in a python form: *wcs['temperature']*. It has also been important to harmonize metadata between services where not harmonized in line with intentions of OWS common in cases where this is actually possible.

The following python code demonstrates some of the harmonized service metadata presented by OWSLib which enables interrogation of the capabilitites of different service types.

```
>>> from owslib.wcs import WebCoverageService
>>> wcs=WebCoverageService('http://somewcs.url.com', version='1.1.0')
>>> wcs.identification.title
' WCS Server from institution A'
```

```
>>> wcs.provider.name
'Institution A'

>>> from owslib.wfs import WebFeatureService
>>> wfs=WebFeatureService('http://somewfs.url.com', version='1.0.0')
>>> wfs.identification.title
' WFS Server from service provider b'
>>> wfs.provider.name
' Service provider B'

>>> from owslib.wms import WebMapService
>>> wms=WebMapService('http://somewms.url.com', version='1.1.1')
>>> wms.identification.title
'WMS Server from a.n.other'
>>> wms.provider.name
' Another service provider'
```

Using the OWSLib WCS module as a WCS client

The process of requesting data from a WCS usually involves the following steps:

1) client makes a GetCapabilitites request to get general information about the server
2) client makes a DescribeCoverage request to get more specific information about a particular coverage
3) client makes a GetCoverage request to get a specific coverage or subset of a coverage

These processes are mirrored in the functionality of the WCS module. However the GetCapabilities and DescribeCoverage requests are not explitly called by the user. They are called 'on-the-fly' by the WCS module when specific metadata is requested. This makes for a simple to use interface.

The following example demonstrates how the WCS module is used in practice.

The first step is to instantiate a WebCoverageService object for a particular WCS service. This will call the GetCapabilities method of the server and populate appropriate python metadata attributes.

```
>>> from owslib.wcs import WebCoverageService
>>> wcs=WebCoverageService('http://somewcs.url.com', version='1.1.0')
>>> wcs.identification.title  #this is one of the service metadata attributes
' WCS Server from institution A'
```

Now find out which coverages are available:

```
>>> wcs.contents
{'AirTemperature': <owslib.coverage.wcs100.ContentMetadata object at 0x8a7fc4c>,
 'WaterVapour': <owslib.coverage.wcs100.ContentMetadata object at 0x8a7fc8c>,
 'SeaSurfaceTemperature': <owslib.coverage.wcs100.ContentMetadata object at
0x8a7fe0c>, ... }
```

It can be seen there are several coverages available. Now find out more about a particular coverage (e.g. its spatio-temporal extent, available output formats). This will silently call the DescribeCoverage method on the server to retrieve coverage specific metadata. DescribeCoverage requests can be expensive (e.g. they may retrieve a long list of available times), so for performance reasons this detailed metadata is only retrieved from the server if it is specificially requested by the client.

```
>>> airtemp=wcs['AirTemperature']  #get the object which represents the metadata of
the AirTemperature coverage
>>> airtemp.boundingBoxWGS84  # get the spatial extent of the coverage in latitude
longitude
   (1.875, -88.767123287700002, 358.125, 88.767123287700002)
>>> airtemp.timelimits #get the temporal extents
   ['2024-01-15T00:00:00.0', '2054-12-15T00:00:00.0']
>>> airtemp.supportedFormats #find out which output formats are supported
   ['cf-netcdf', 'GeoTiff']
```

By using the information gained via such interogation a GetCoverage request may be formulated and sent to the server:
output=wcs.getCoverage(identifier='AirTemperature', time=['2024-01-15T00:00:00.0'], bbox=(-80,30,50,60), format='cf-netcdf')

(Note that other request options are available but this example has been somewhat simplified for brevity.)

The output coverage file may then be written to disk and viewed using suitable software:

```
>>> f=open('test.nc', 'wb')
>>> f.write(output.read())
>>> f.close()
```

This code has shown a command-line python example, but the OWSLib module could equally be incorporated into a desktop or web-based client as the WCS client 'engine'.

Very Rudimentary Exercise of a Few GALEON WCS Servers

It is tempting to say that the OWSLib Python library has been used to conduct a set of interoperability experiments or at least tested a set of WCS servers, but the reality is that the initial work has been confined to a set of simple exercises of a set of GALEON servers that illustrate the usefulness and potential of OWSLib and show that a set of GALEON servers can deliver coverages via the WCS 1.0 and 1.1 protocols. The exercises are embodied in Python scripts that do the following:

- Use getCapabilities for list of coverages
- Extract name of first coverage in list
- If multiple times are available, use the initial time
- Use getCoverage to request first coverage
- Use defaults wherever possible

on WCS servers at the following sites:

- US Pacific Fisheries Environmental Lab
- US Unidata THREDDS Data Server
- US USGS Coast Environmental
- US NCDC NOMADS
- Neitherlands KNMI Geoservices
- Netherlands ARGOSS
- UK BADC
- US NSIDC
- US George Mason U.
- Italy U. of Florence CNR
- US NNEW Weather.aero

General Results

The initial python script runs resulted in a mixed set of results:

- 3 servers returned a coverage on the first try
- 2 required minor changes to the default parameter list
- The rest required interactions between Dominic, Ben, and the WCS site administrator
- 2 cases of missing required parameters
- 2 cases uncovered bugs in python client library which were subsequently fixed

After a few interactions with the administrators of the WCS servers, all cases working in the end. The interactions are documented on a GALEON team wiki page: https://sites.google.com/site/galeonteam/Home/GALEON%20WCS%20"Tests"

Conclusion and Future Enhancements

These exercises showed conclusively that the OWSLib client libraries provide a relatively simple and straightforward means for constructing clients that can be used effectively to exercise WCS servers. As the number of available servers increases, the scripts can be enhanced so they become something closer to a test of the servers. To do so, the GALEON team will have to:

- Include more servers in the exercise

- Augment exercise in the direction of a test
- Retrieve representative coverages
- Retrieve representative times
- Use bounding box other than default
- Examine returned coverage
- Track the evolution of WCS specification

More Information

- Acronym Glossary: http://www.unidata.ucar.edu/content/publications/acronyms/glossary.html
- OWSLib wiki: http://trac.gispython.org/lab/wiki/OwsLib
- Subversion Repository: svn co http://svn.gispython.org/svn/gispy/OWSLib/trunk
- Python Cheeseshop: http://pypi.python.org/pypi/OWSLib/0.3
- GALEON Wiki page: https://sites.google.com/site/galeonteam/Home/GALEON%20WCS%20"Tests"
- CF-netCDF binary encoding profile for WCS. http://www.unidata.ucar.edu/projects/THREDDS/GALEON/CF-netCDFprofile/08-XXX_WCS_Extension_Standard_for_CF-netCDF_Coverage_Encoding.doc
- **GALEON Phase 1 Summary Report** (http://www.unidata.ucar.edu/projects/THREDDS/GALEON/Reports/GALEONphase1Report.htm)
- LEAD (Linked Environments for Atmospheric Discovery): http://lead.ou.edu
- Proposed CF Conventions for Point Data (proposed) http://www.unidata.ucar.edu/software/netcdf-java/CDM/CFpoints.html
  "Unidata's Common Data Model Mapping to the ISO 19123 Data Model" http://www.springerlink.com/content/t5g828928v82n6ju/, S. Nativi, J. Caron, B. Domenico, L. Bigagli, , Earth Sci Inform DOI 10.1007/s12145-008-0011-6, Springer-Verlag 2008.
- WCS-THREDDS Gateway: http://my.unidata.ucar.edu/content/projects/THREDDS/OGC/WCS-THREDDS%20Gateway.htm