

NextGen Network Enabled Weather (NNEW)
Federal Fiscal Year 2009
Information Technology Demonstration
Implementation Verification Procedures



Version 1.6

9/15/2009

National Center for Atmospheric Research

MIT Lincoln Laboratory

National Oceanic and Atmospheric Administration/Global Systems Division

DOCUMENT REVISION REGISTER

Version	Date	Content Changes	Editors	Contributors
0.1	08/17/09	Initial template with new input for registry/repository		Farrukh Najmi
0.2	8/20/09	Addition of WFS-RI, Security, QoS, and Ontology Alignment verification	Arnaud Dumont	Kajal Claypool
0.3	8/27/09	Addition of WCS-RI implementation verification	Arnaud Dumont	Rob Weingruber
0.4	8/29/09	Addition of Flight Hazard and Higher-Order Service composition verification	Arnaud Dumont	Marcel Casado
1.1	9/9/09	Final revision of Registry Repository		Farrukh Najmi
1.2	9/9/09	Final addition of NWS section	Arnaud Dumont	Matthew Peroutka Steve Olson
1.3	9/11/09	Final edits of WCS-RI section	Arnaud Dumont	Rob Weingruber
1.4	9/13/09	Final additions to WFSRI and Ontology	Arnaud Dumont	Kajal Claypool
1.5	9/13/09	Final additions to Flight Hazard Services	Arnaud Dumont	Marcel Casado
1.6	9/15/09	Final edits and formatting for dry run and including comments and edits of run	Aaron Braeckel Arnaud Dumont	Rob Weingruber Marcel Casado

Please direct comments or questions to the contributors of the sections listed above.

For overall comments or questions, contact:

Arnaud Dumont
 National Center for Atmospheric Research
 Research Applications Laboratory
 3450 Mitchell Lane
 Boulder, CO 80301
dumont@ucar.edu
 (303)497-8434

TABLE OF CONTENTS

1.	Overview	7
2.	Test Requirements	8
2.1	System Dependencies	8
2.2	Demonstration Applications	9
3.	Implementation Verification - Registry / Repository	12
3.1	Test Environment and Setup	12
3.1.1	Client Software	12
3.1.2	Registry and Federation Setup	12
3.1.3	Initial Data Setup	13
3.1.4	Launching the Registry Administration UI version 4.4	14
3.1.5	Launching the Registry Administration UI version 4.5-SNAPSHOT	14
3.2	Discovery of Dataset by Weather Phenomenon Type	15
3.2.1	Dataset Discovery - Unfiltered Local Search	15
3.2.2	Dataset Discovery - Unfiltered Federated Search	17
3.2.3	Dataset Discovery - Local Search Filtered By Dataset Field	18
3.2.4	Dataset Discovery - Federated Search Filtered By Dataset Field	21
3.2.5	Dataset Discovery - Semantic Filtered Search	23
3.2.6	Dataset Discovery – REST Search Filtered By Dataset Field	27
3.3	Discovery of Datasets by Weather Cube Domain Classification	30
3.3.1	Viewing an Existing Taxonomy	31
3.3.2	Dataset Discovery - Local Search Filtered By Any Weather Cube Domain	32
3.3.3	Dataset Discovery - Local Search Filtered By “Unrestricted” Weather Cube Domain	34
3.4	Discovery of Services Instances	35
3.4.1	Service Discovery - Unfiltered Local Search	36
3.4.2	Service Discovery - Unfiltered Federated Search	37
3.4.3	Service Discovery - Local Search Filtered By Service Type	38
3.4.4	Service Discovery - Federated Search Filtered By Service Type	43
3.4.5	Service Endpoint Retrieval	45
3.4.6	Viewing Datasets Related to a Service	47
3.5	Creation of an Experimental Weather Cube Taxonomy	49

3.6	Publication of an Experimental Data Set and Accompanying Experimental Data Access Service	52
3.6.1	Publish Dataset	52
3.6.2	Publish Service Instance	55
3.7	Fault Tolerance Support in Registry Client API	63
4.	Implementation Verification – Web Coverage Service Reference Implementation (WCSRI)	65
4.1	Test Environment and Setup	66
4.1.1	Dependency Installations	66
4.2	WCSRI Administration	68
4.2.1	Services Metadata	69
4.2.2	Configuring a New Coverage	70
4.2.3	Starting Fuse/ServiceMix	72
4.3	WSDL Verification	74
4.4	Verification using Maven and Black Box Testing	75
4.5	Verification using SoapUI GUI and ToolsUI	76
4.5.1	Starting SoapUI and Configuring the WCSRI Endpoint	76
4.5.2	Executing the Black Box Test Suite	79
4.5.3	Executing GetCapabilities	79
4.5.4	Executing DescribeCoverage	81
4.5.5	Executing GetCoverage for a Volume	81
4.5.6	Visualizing GetCoverage Volume Results	83
4.5.7	Executing GetCoverage for a Corridor	86
4.5.8	Visualizing GetCoverage Corridor Results	89
4.6	Verification with NNEW FY09 Integrated Java Application	91
5.	Implementation Verification – Web Feature Service Reference Implementation (WFSRI)	97
5.1	Test Environment and Setup	101
5.1.1	Administration Client	101
5.1.2	Publish Clients	101
5.1.3	Retrieval Clients	103
5.1.4	CIWS Display	104
5.2	Register New Producer Using the WFSRI Administrator	105

5.3	Create Feature Table	106
5.4	Register Feature Type Using the WFSRI Administrator	107
5.5	Transaction Insert	109
5.5.1	Using the Generic Client	109
5.5.2	Using the ATOM-WFS Bridge	111
5.6	GetCapabilities Using the Generic Client	112
5.7	DescribeFeatureType Using the Generic Client	113
5.8	GetFeature Request/Response	114
5.8.1	Unfiltered Access	115
5.8.2	Spatial Subsetting	116
5.8.3	Temporal Subsetting	116
5.9	GetFeature Subscription Using Google Earth Subscription Client	117
5.9.1	Unfiltered Subscription	117
5.9.2	Spatial Filtering Subscription	119
5.10	Security	119
5.10.1	Verification of Unauthorized Access	119
5.10.2	Verification of Authorized Access	120
5.11	Client-Side Service Adaptor – CIWS Display	120
5.11.1	Precip Product	121
5.12	Winter Precip Product	121
5.13	Echo Tops Product	122
5.14	Lightning Product	123
5.15	AIXM Overlays	124
6.	NWS Data discovery and access	126
6.1	Introduction	126
6.2	Test Environment and Setup	126
6.3	Discovery of NDFD/NDGD SOAP Service	126
6.4	SOAP Query of NDFD/NDGD Data	128
6.5	Web Coverage Service Query of NDFD/NDGD Data	133

7.	Implementation Verification – Flight Hazard Service and High Level Service Capability	140
7.1	Test Environment and Setup	140
7.1.1	Client Software	140
7.1.2	Composite Services Setup	141
7.2	Verification of 4D Trajectory Capabilities and Flight Hazard Service Composition with Flight Hazard Tool Web UI	142
7.2.1	Weather Hazards Along a Flight Trajectory - Archived Use Case	142
7.2.2	Weather Hazards Along a Flight Trajectory – “Near Future” Use Case	149
7.3	Verification of Flight Hazard Composition Service and Composite Services with SoapUI150	
7.3.1	Atomic Services Test Steps	150
7.3.2	Flight Hazard Service Composition Test Case	159
7.3.3	Flight Hazards Service – Black Box Test Case	163
7.4	Verification Derived Data with ToolsUI	164
8.	Ontology Alignment Tool	168
8.1	Test Environment and Setup	168
8.1.1	Ontology Alignment UI	168
8.2	Loading Ontology	168
8.3	Manual Alignment of Ontologies	170
8.4	Semi-Automated Alignment of Ontologies	171
8.5	Loading an Existing Alignment	173
9.	References	176

1. Overview

This document provides procedures for verification of work completed for NNEW during the federal fiscal year 2009 (FY '09). In many cases, these capabilities are extensions to the capabilities developed for NNEW in previous years, 2007-2008. Specific mention will be made when appropriate.

NNEW implementation during FY '09 was focused primarily on 5 areas: implementing federated registry/repositories, developing a Web Coverage Service Reference Implementation (WCSRI), developing a Web Feature Service Reference Implementation (WFSRI), orchestrating services in the Fuse ESB container mandated by the FAA's System-Wide Information Management (SWIM) program, and developing an initial security infrastructure for managing data access.

2. Test Requirements

The verification procedures outlined in this document may be executed on any computer that satisfies a minimum set of requirements. System requirements include basic system dependencies and demonstration applications.

System dependencies can be satisfied by installing publicly available components, typically released by parties unaffiliated with the NNEW project. In all cases, these components are available without licensing costs.

Demonstration applications are specialized tools that facilitate interaction with NNEW standard services or formats. In many cases, these applications were created by NNEW developers primarily for the purpose of this test. Those tools may be unavailable or unsupported after the test is complete.

2.1 System Dependencies

System dependencies are required to run the demonstration applications. These include computer memory, network connectivity, libraries, and utility packages. All of the items below must be installed prior to installing and configuring the demonstration applications:

- **Memory**

The test computer should have a minimum of 1048M of RAM. This memory is primarily needed for responsiveness when running the NNEW FY '09 IT Demonstration Integrated Java Application. Performance for visualization of some of the larger datasets is greatly improved for clients with 512M or 1G of RAM.

- **Internet Connection**

The test computer must be able to initiate an outgoing TCP/IP connection to port 80 of a remote host. The test computer must also be able to receive data across the connection it initiates. This fundamental capability can be verified using any commercial off-the-shelf web browser.

The test computer may be connected via a proxy server and/or have its data filtered through a web filter, as long as the intervening software does not corrupt digital signatures on data. Some web filters have been known to invalidate the certificate signatures on valid data when they attempt to inspect its contents. If you encounter corrupted certificate errors when running the tests, please contact your system administrator for assistance.

- **Java Web Start**

The test computer must have the latest stable version of Java installed. This is currently Java 1.6.0_16 (eg: build_1.6.0_16-b03). Java may be downloaded and installed by pointing a web browser to the following URL:

<http://java.com>

Installation requires administrator privileges on most platforms.

The latest version of Java includes Java Web Start. Java Web Start is a client-side Java Virtual Machine environment that facilitates execution of code downloaded from the internet. Java Web

Start provides many security protections not available when running applications directly. It also allows signed applications to request access to local printers, create and load files on the local file system, initiate outgoing TCP/IP connections, etc. The NNEW FY '09 IT Demonstration Integrated Java Application and Catalog/Registry Interface Application both require Java Web Start.

- **Maven**

The Maven exec plugin is required by the SoapUI application.

- **Subversion**

Subversion is required to download source code and components from the NNEW code repository.

- **Fuse ESB 4.1.0.2 (ServiceMix)**

The Fuse ESB is the platform on which the WCSRI and WFSRI services are hosted. This ESB is required by the FAA System-Wide Information Management (SWIM) program.

- **NetCDF 4 C Libraries**

The NetCDF 4 C libraries must be installed and available via the `$LD_LIBRARY_PATH` variable when running the Fuse ESB. The NetCDF 4 libraries are used by the MIT NetCDF JNI libraries, described below.

- **HDF5**

The HDF5 library is required by the NetCDF 4 libraries. It is available for download from the NetCDF web site.

- **MIT Lincoln Labs NetCDF JNI Libraries**

The Lincoln Labs NetCDF JNI C libraries must be built, installed and available via the `$LD_LIBRARY_PATH` variable when running the Fuse ESB Demonstration Applications.

2.2 Demonstration Applications

Several applications were developed or modified to demonstrate the FY '09 implementation. These applications were created because there is no way for a person to interact with NNEW system using existing, off-the-shelf, software; security requires authentication of clients, access services adhere to rigid protocols for information exchange, and returned datasets are typically encoded in compact binary formats. In addition, the 4D Weather Functional Requirements for NextGen Air Traffic Management specifically calls for "Weather integrated directly into sophisticated decision-support capabilities." That requirement clearly pushes development focus towards machine-to-machine interoperability and away from human interfaces. As a result, the NNEW team has chosen to avoid documenting the interactions needed for users to test the implementation directly, and instead, provides custom applications that test the implementation.

The demonstration applications range from simple, single-use, tools to complex, integrated, displays. In addition to verifying the implementation, these applications demonstrate the flexibility of the operational

concept: many types of clients, interacting with the system using several different protocols, and requesting data at many different granularities. The demonstration applications are:

- **NNEW FY '09 Integrated Java Application**

This is an extension of the Phase II demonstration application. It integrates most of the products produced for the NNEW FY '09 demonstration. This tool demonstrates the use of the service registries, gridded data services, and feature services from all three NNEW laboratories.

- **Catalog/Registry User Interface Application**

This tool connects to a registry and allows for interactive uploading, querying, and downloading of service information (WSDL's and associated schemas), dataset metadata, classification taxonomies, and other artifacts associated with geospatial data artifacts, such as coordinate reference system dictionaries. The UI may be launched by pointing your web browser to the following URL:

<http://www.wellfleetsoftware.com/files/ui/4.4/jnlp/wellgeo-ui-swing.jnlp>

- **WFSRI Administration User Interface**

This is the user interface for administration of the Web Feature Service Reference Implementation.

- **Ontology Alignment User Interface**

This application facilitates the alignment of ontologies.

- **Firefox Web Browser**

A standards-compliant web browser is required to run several of the web applications and web user interfaces. While these tests may run on non-standards-compliant browsers, user satisfaction may be impacted. To ensure that graphical layout, event handling, and performance are optimized, the latest stable release of Mozilla Firefox is highly recommended. This application may be downloaded from:

<http://www.firefox.com>

- **SoapUI 2.5.1 and 3.0**

SoapUI is an application that provides viewing and invocation of SOAP XML requests, as well as viewing of SOAP XML responses. Version 2.5.1 is required for Section 7, Implementation Verification – Flight Hazard Service and High Level Service Capability. Version 3.0 is required for all other sections that use SoapUI. Both versions may be installed on a test system in parallel.

- **Ncdump**

The ncdump application is a simple script for printing out the data structures of NetCDF files. It is used in this verification as a way to confirm the metadata of weather variables in returned data files (their name, dependent dimensions, units, etc). Ncdump comes bundled with the NetCDF library. The library can be downloaded from the following URL (please ensure that the downloaded libraries support NetCDF version 4):

<http://www.unidata.ucar.edu/software/netcdf>

- **ToolsUI**

The ToolsUI application is a Java Web Start application that provides viewing of the data structures of NetCDF files. The version of ToolsUI must support NetCDF 4.

3. Implementation Verification - Registry / Repository

The NNEW registry is used to store high-level information about datasets and their associated services. Service interface information is used at build-time to generate client applications that conform to a given interface. At run time, a common usage model for the registry is to discover datasets using concepts such as 'air_temperature', and then discover the service(s) capable of serving those datasets. The ability to discover datasets classified as members of the NextGen Single Authoritative Source (SAS) for weather is also an important use case.

The NNEW registry is deployed as a federated registry consisting of a group of individual registry instances, each operated independently by a different NNEW member Organization. Interoperability between different registries in the federation is enabled by the OASIS ebXML RegRep 4 specifications.

The test cases in this chapter cover these core build-time and run-time usage scenarios.

3.1 Test Environment and Setup

3.1.1 Client Software

The series of registry tests defined in this document use the following registry client software:

- A Java-based Registry Administration UI – This registry client provides a UI that enables publish and discovery of datasets, service instances, service interfaces, taxonomies and more. Its discovery features allow searching individual registries using local searches or searching registry federations consisting of multiple registries using federated searches.
- A [RegistryTestClient](#) command line program - This registry client is a test program that is used to test the fault-tolerance and client-side support for federated queries within the registry client API library [regrep4-client](#) .

In addition to above clients, the NNEW Phase Integrated Java Application is also used as a user-oriented registry client application within this test plan.

3.1.2 Registry and Federation Setup

The 2009 test plan defines the following registry and federation setup:

- NNEW Federation 1 - A federation with the following registries
 - Lincoln Labs Registry 1 - A registry actively operated by MIT Lincoln Labs. Abbreviated as “MIT-LL registry”.
 - NWS Registry 1 – A registry actively operated by National Weather Service. Abbreviated as “NWS Registry”
 - GSD Registry 1 – A registry operated by NOAA / GSD. This registry is not actually active for the 2009 Test plan and exists only to test fault-tolerance aspects of federated search.

Exact configuration for above federation [is available as an eBRIM file](#)

- FAA Technology Center 1 – A registry actively operated by the FAA Technology Center. This registry will be used a backup registry for the registries in NNEW Federation for the registry client API fault-tolerance tests. Abbreviated as “FAA Registry”.

3.1.3 Initial Data Setup

At the start of the test, each registry is pre-loaded with the following types of information:

- **Dataset descriptions** for datasets provided by the organization that operates the registry. A Dataset description consists of a file formatted to the ISO 19139 metadata standard, a worldwide standard that is slated to replace the FGDC standard in the U.S. in the years ahead. The following datasets are pre-loaded into each registry:
 - Datasets in MIT-LL Registry:
 - [MIT-LL Datasets](#): Vertically Integrated Liquid (VIL), EchoTops, Lightning
 - [NCAR Datasets](#): CEIL, CIP-20, CIPSEV-20, FLTCAT, GTG2, METARS-1, PIREPS-1, RUC20_Air_Temperature, RUC20_Relative_Humidity, RUC20_Wind, VIS
 - [DoD Datasets](#): DOD_Air_Temperature
 - Dataset in NWS Registry:
 - [NWS Datasets](#): WindDirection-NDFD-CONU, WindSpeed-NDFD-CONUS
 - [GSD Datasets](#): MDCRS-1, METARS-1, PIREPS-1, RR_Air_Temperature-1, RR_Wind-1, Surface_Air_Temperature-1, Surface_Dewpoint-1, Surface_Wind-1
- **Service descriptions** for the services provided by the organization that operates that registry. Service information is specified as a combination of ISO 19139 metadata and W3C WSDL files. The following service descriptions are pre-loaded into each registry:
 - Service in MIT-LL Registry:
 - [MIT-LL Services](#): MIT WCS-01/02/03, MIT WFS
 - [NCAR Services](#): NCAR WCS, NCAR WFS-01
 - [DoD Services](#): DOD-JMBL-01
 - Services in NWS Registry:
 - [NWS Services](#): NDFD-WFS-01
 - [GSD Services](#): NOAA-WCS-01, NOAA-WCS-02, NOAA-WFS-01

- **Taxonomies and other configuration** for each registry. These include any taxonomies, object types, association types and other configuration that are required by the NNEW registry profile. An example is the [weather cube taxonomy](#). These pre-loaded objects are defined using the ebXML RegRep 4 Registry Information Model schema.

Note that all test metadata is defined in the [wxcube-metadata](#) project at WxForge. The hyperlinks above link to organization specific directories within the project. Testers MUST checkout the wxcube-metadata project in a test directory (referred to as \$TEST_DATA_DIR in this plan).

3.1.4 Launching the Registry Administration UI version 4.4

For most tests we will be using the Registry Administration UI version 4.4. This version of the UI may be launched by one of the following methods:

- Point your web browser to the following URL:
<http://www.wellfleetsoftware.com/files/ui/4.4/jnlp/wellgeo-ui-swing.jnlp>
- Use the javaws program as follows:
javaws <http://www.wellfleetsoftware.com/files/ui/4.4/jnlp/wellgeo-ui-swing.jnlp>

This will launch a Registry Administration UI version 4.4 that has the various NNEW registries configured. By default, the admin UI will connect to the MIT-LL registry. To select another registry (e.g. NWS) you may select the URL for that registry in the “Registry Base URL” field of the “Options” dialog which is access via the Tools / Option menubar action. Note that switching registry URL may take up to a minute and no busy indicator is shown during the action.

3.1.5 Launching the Registry Administration UI version 4.5-SNAPSHOT

For federated search tests we will be using the Registry Administration UI version 4.5-SNAPSHOT¹. This version of the UI may be launched by one of the following methods:

- Point your web browser to the following URL:
<http://www.wellfleetsoftware.com/files/ui/4.5-nnew/jnlp/wellgeo-ui-swing.jnlp>
- Use the javaws program as follows:
javaws <http://www.wellfleetsoftware.com/files/ui/4.5-nnew/jnlp/wellgeo-ui-swing.jnlp>

This will launch a Registry Administration UI version 4.5-SNAPSHOT that also has the various NNEW registries configured. By default, the admin UI will connect to the MIT-LL registry. To select another registry (e.g. NWS) you may select the URL for that registry in the “Registry Base URL” field of the “Options” dialog which is access via the Tools / Option menubar action. Note that switching registry URL may take up to a minute and no busy indicator is shown during the action.

¹ The reason for using a newer version of the UI is that it shows the name of the source registry for objects in federated search results

3.2 Discovery of Dataset by Weather Phenomenon Type

This test demonstrates the query capabilities of the Registry/Repository. In particular, it demonstrates:

1. Unfiltered Search: Discovery of all registered datasets without specifying any specific filter criteria.
2. Filtered Search: Discovery of all registered datasets filtered by a particular weather phenomenon type. The filter provides an exact match or a regular expression match to the specified weather phenomenon.
3. Semantic Filtered Search: Discovery of all registered datasets filtered by a particular weather phenomenon type and other phenomenon *similar* to it. The filter in this case expands the specified weather phenomenon to all similar phenomenon using an ontology.
4. Federated Search: Discovery of datasets within multiple registries in a registry federation.

3.2.1 Dataset Discovery - Unfiltered Local Search

This test performs an unfiltered search for all registered *data sets* in the MIT-LL Registry.

1. Launch the Registry Admin UI **version 4.4** as described in section 3.1.4. By default it will connect with the MIT-LL registry.
2. Click on the Search tab in upper left corner of UI to access the Search Tool panel.
3. Make sure that the Federated Query Options combo box shows “**Local Query**” as selection
4. Select “**Find Data Set**” in the Select Query combo box.
5. Click the “**Search**” button at the top of the Search Tool panel. An unfiltered list of all the registered datasets will appear. Click on the “**Name**” column heading to sort the datasets by name / Name (alphabetical order).
6. Verify that the list of datasets returned includes the following:
 - Current Icing Potential
 - Current Icing Potential
 - Current Icing Severity
 - DoD Model Air Temperature
 - Echo Tops (Experimental)
 - Echo Tops (Pending SAS)
 - Echo Tops (SAS)
 - Echo Tops (SAS Backup)
 - Flight Category at Surface
 - Geometric Height at Cloud Base

- Lightning
- METARS
- National Convective Weather Forecast Model
- PIREPS (ADDS Source)
- RUC Model - 20 kilometer resolution
- Turbulence
- Vertically Integrated Liquid (VIL)
- Visibility at Surface

Note: The sorting based on Name is transient and does not currently remain in effect from query-to-query. This has been submitted as a feature request for the UI.

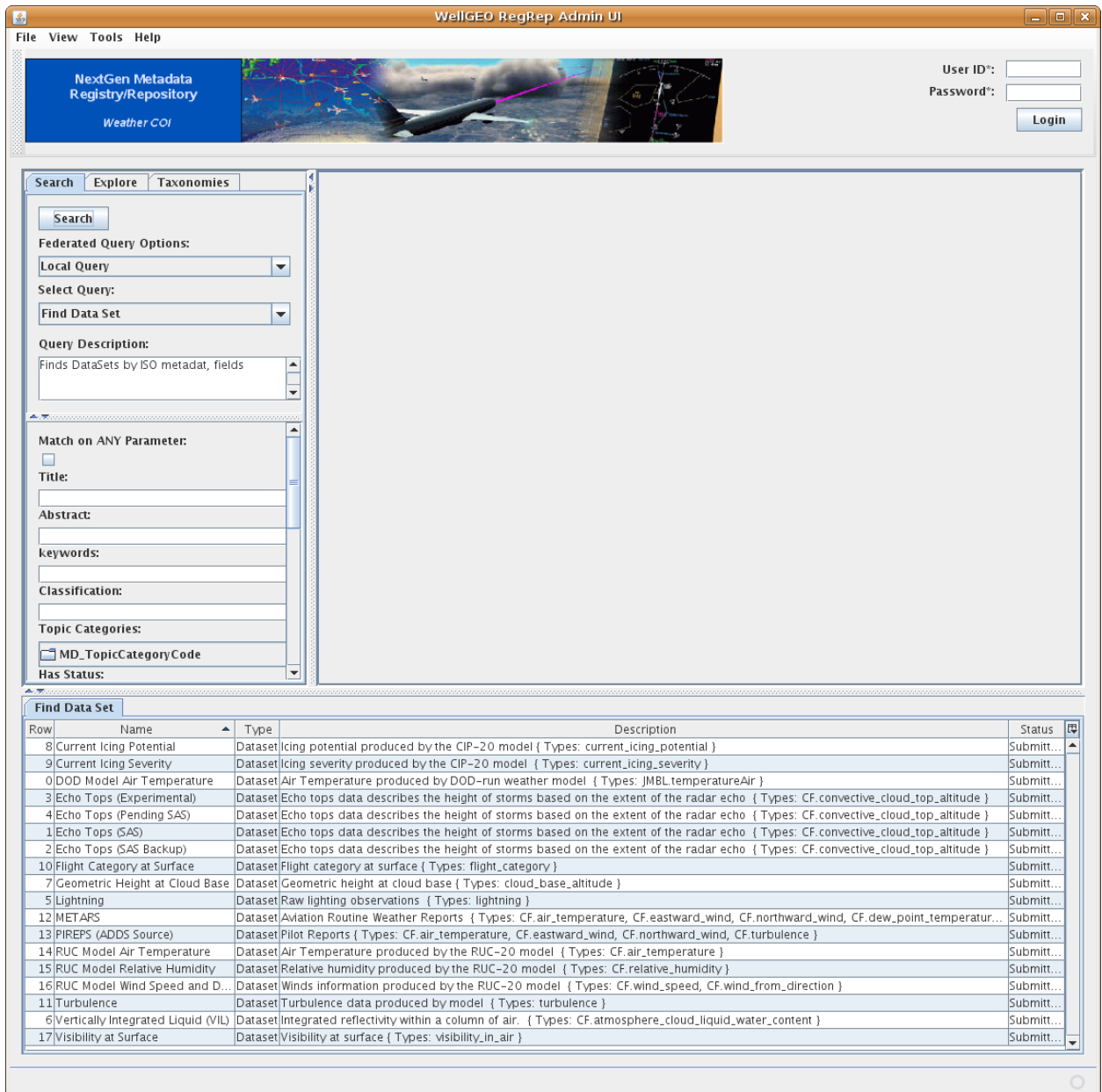


Figure 3.1: Dataset Discovery - Unfiltered Local Search: Shows datasets from MIT-LL registry

3.2.2 Dataset Discovery - Unfiltered Federated Search

This test performs an unfiltered search for all registered *data sets* across all registries in the NNEW Federation 1 which currently includes MIT-LL and NWS registries.

1. Launch the Registry Admin UI **version 4.5-SNAPSHOT** as described in section 3.1.5.
2. Repeat the previous test “Dataset Discovery - Unfiltered Local Search” with one change; Make sure that in step 3, the Federated Query Options combo box shows “**NNEW Federation 1**” as selection.

- Verify that the list of datasets returned includes datasets from both the MIT-LL and NWS registries as indicated by the last column labeled “Registry” in search result.

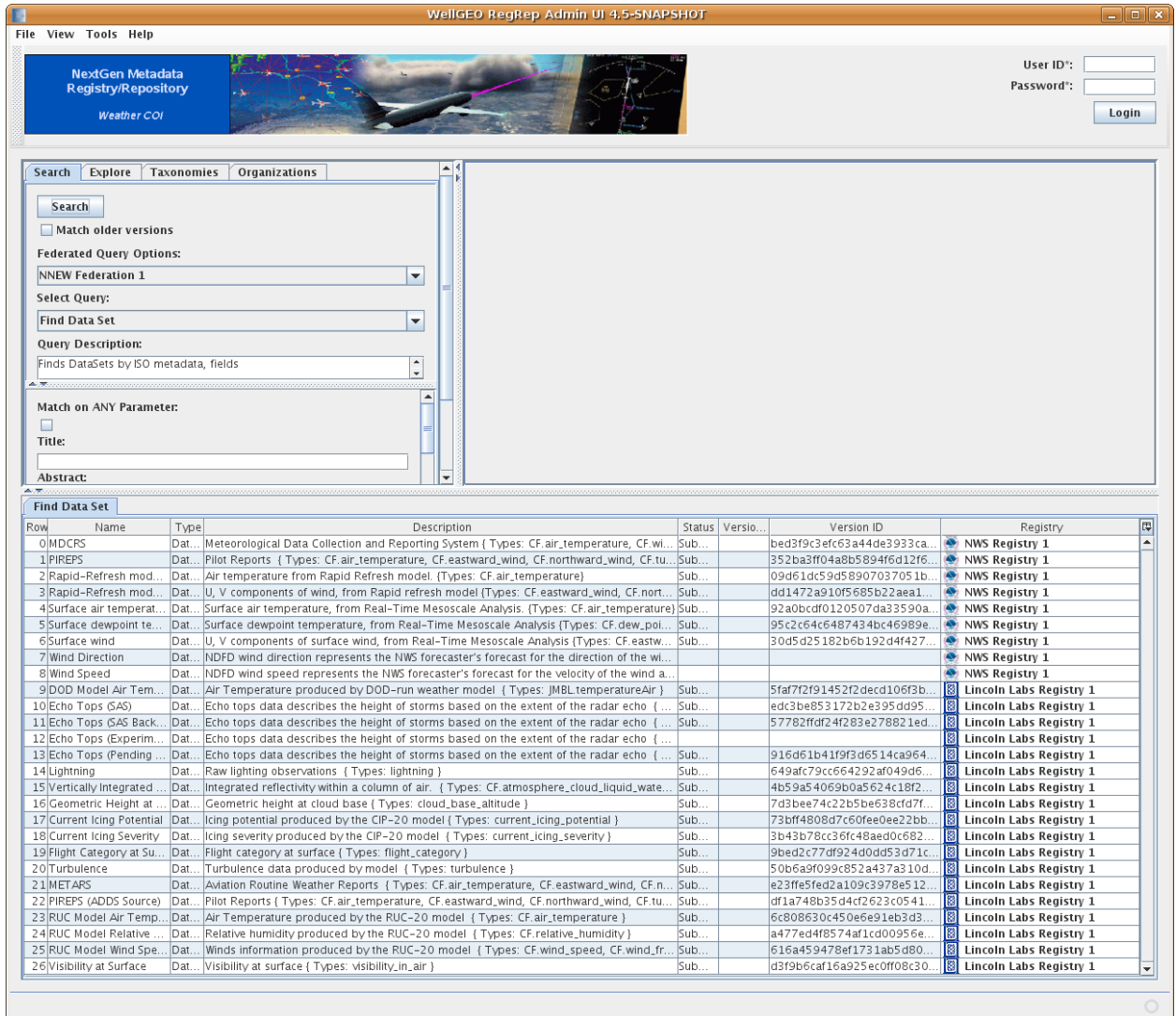


Figure 3.2: Dataset Discovery - Unfiltered Federated Search
Shows datasets from MIT-LL and NWS Registries

3.2.3 Dataset Discovery - Local Search Filtered By Dataset Field

This test performs a filtered search for all data sets within MIT-LL registry with the weather phenomenon type *air_temperature*, where *air_temperature* is a Climate and Forecast (CF) conventions term; and weather phenomenon type *temperatureAir*, where *temperatureAir* is a Joint METOC Broker Language (JMBL) term.

- If necessary, launch the Registry Admin UI **version 4.4** as described in section 3.1.4. By default it will connect with the MIT-LL registry.
- Click on the “Search” tab in upper left corner of UI to access the Search Tool panel.

3. Make sure that the Federated Query Options combo box shows “**Local Query**” as selection
4. Select “**Find Data Set**” in the Select Query combo box.
5. Type the following text in the “**Dataset Field**”:
`air_temperature`
6. Click the “**Search**” button. A filtered set of all registered datasets with weather phenomenon type of *air_temperature* are returned. Click on the “**Name**” column heading to sort the datasets by name / Name (alphabetical order).
7. Observe the matching data sets that are returned include:
 1. METARS
 2. PIREPS (ADDS Source)
 3. RUC Model - 20 kilometer resolution

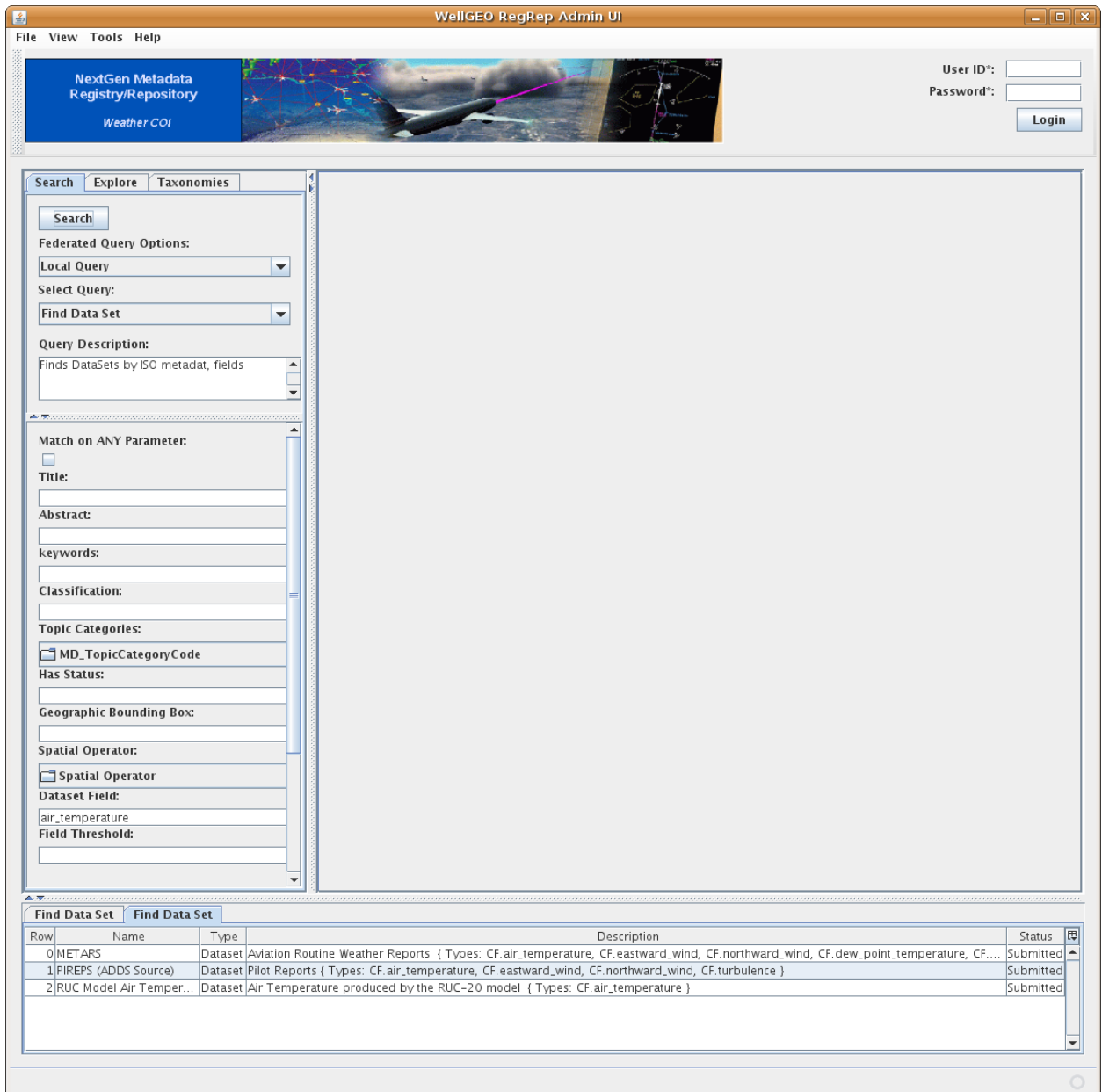


Figure 3.3: Dataset Discovery - Local Search Filtered By Dataset Field:
Shows filtered results from MIT-LL registry matching Data Set Field = air_temperature

7. Repeat the search only this time type the following text in the “**Data Set Field**”:
temperatureAir
8. Click the “**Search**” button. A filtered set of all registered datasets with weather phenomenon type of *temperatureAir* are returned.
9. Verify that the single matching data set that is returned is:
 - DOD Model Air Temperature

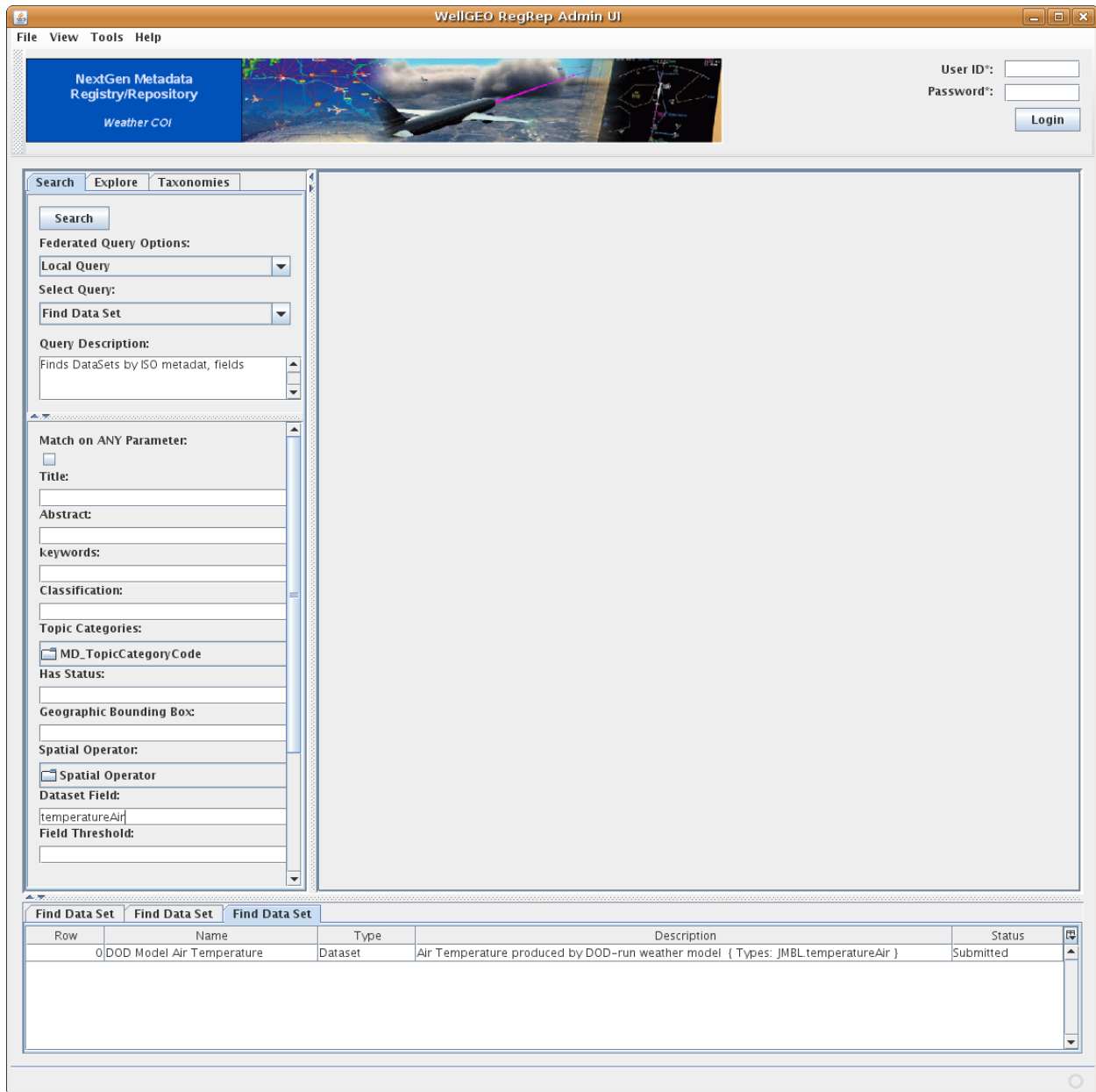


Figure 3.4: Dataset Discovery - Local Search Filtered By Dataset Field:
Shows filtered results from MIT-LL registry matching Data Set Field = temperatureAir

3.2.4 Dataset Discovery - Federated Search Filtered By Dataset Field

This test performs a filtered search for all data sets across all registries in the NNEW Federation 1 which currently includes MIT-LL and NWS registries. The filter is the same as previous test with the weather phenomenon type *air_temperature*, where *air_temperature* is a Climate and Forecast (CF) conventions term; and weather phenomenon type *temperatureAir*, where *temperatureAir* is a Joint METOC Broker Language (JMBL) term.

1. Launch the Registry Admin UI **version 4.5-SNAPSHOT** as described in section 3.1.5.

2. Repeat the previous test “Dataset Discovery - Local Search” with one change; Make sure that in step 3, the Federated Query Options combo box shows “**NNEW Federation 1**” as selection.

Verify that the list of datasets returned in step 7 includes datasets from both the MIT-LL and NWS registries as indicated by the last column labeled “**Registry**” in search result:

- MDCRS
- PIREPS
- Rapid-Refresh model air temperature
- Surface air temperature
- METARS
- PIREPS (ADDS Source)
- RUC Model - 20 kilometer resolution

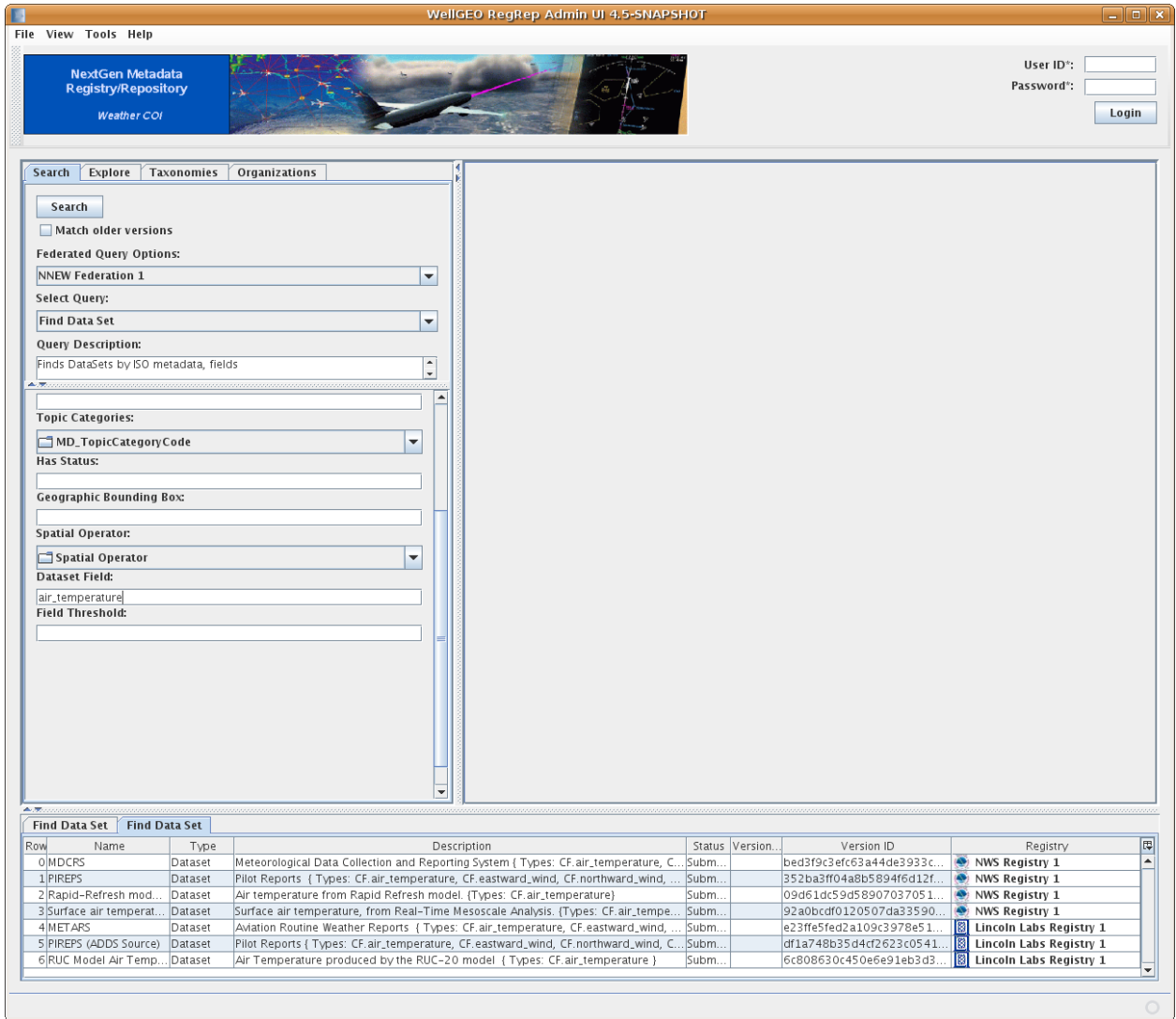


Figure 3.5: Dataset Discovery - Federated Search Filtered By Dataset Field
Shows filtered results from MIT-LL and NWS registries matching Data Set Field = temperatureAir

3.2.5 Dataset Discovery - Semantic Filtered Search

This test performs a semantically-enhanced filtered search for all data sets with the weather phenomenon type *temperatureAir*, where *temperatureAir* is a Joint METOC Broker Language term; *air_temperature*, where *air_temperature* is a Climate and Forecast conventions term; and *CloudLiquidWater*, a Joint METOC Broker Language term for vertically integrated liquid. The Climate and Forecast term equivalent to *CloudLiquidWater* is *atmosphere_cloud_liquid_water_content*. The test uses the weather phenomenon ontology, Climate and Forecast, and Joint METOC Broker Language taxonomies, and alignments between these different entities.

Launch the Registry Admin UI **version 4.4** as described in section 3.1.4. By default it will connect with the MIT-LL registry.

1. Click on the “**Search**” tab in upper left corner of UI to access the Search Tool panel.
2. Make sure that the Federated Query Options combo box shows “**Local Query**” as selection
3. Select “**Find Data Set**” in the Select Query combo box.

4. Type the following text in the “**Data Set Field**”:

`temperatureAir`

5. Type the following value in the “**Data Set Threshold**”:

`0.5`

The threshold indicates that data sets whose weather phenomenon type matches the specified weather phenomenon type at the threshold value or above are returned to the user. The threshold takes a value in the range of 0.0 to 1.0. Set the threshold to 0.5 to pick up any datasets whose weather phenomenon type matches *temperatureAir* with a 0.5 confidence or higher.

6. Click the “**Search**” button. A filtered set of all registered datasets with weather phenomenon type of *temperatureAir* and similar (confidence > 0.5) weather phenomenon types are returned. Click on the “**Name**” column heading to sort the datasets by Name (alphabetical order).
7. Verify the matching data sets that are returned are:
 - DOD Model Air Temperature
 - METARS
 - PIREPS (ADDS Source)
 - RUC Model – 20 kilometer resolution

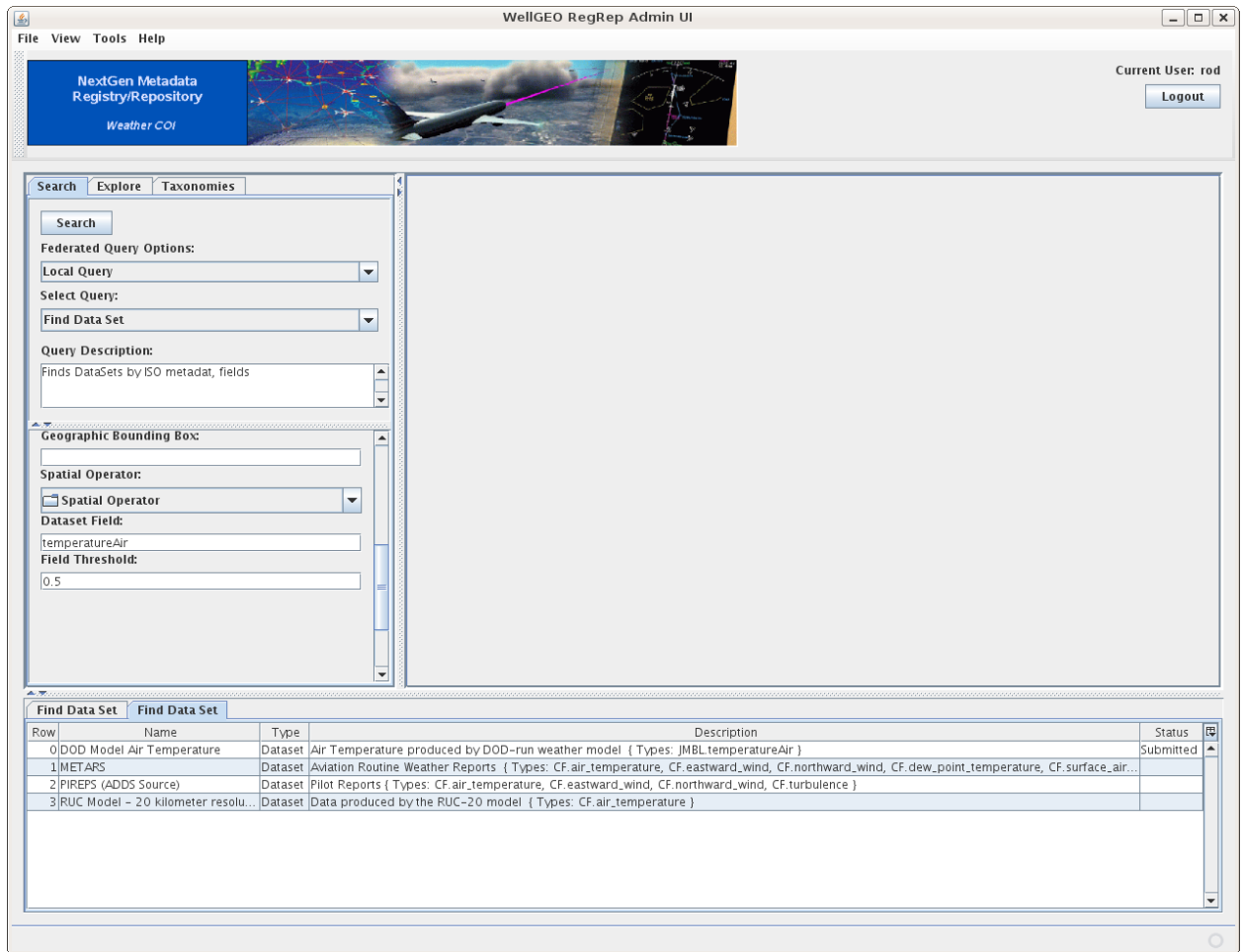


Figure 3.6: Snapshot of the Registry UI showing the results of semantic filtered search, with Data Set Field = temperatureAir, and Data Set Threshold = 0.5.

8. Repeat the **Search for Data Set** (Step 4). Type the following text in the “**Data Set Field**”:
air_temperature
9. Type the following value in the “**Data Set Threshold**”:
0.5
10. Click the “**Search**” button. A filtered set of all registered datasets with weather phenomenon type of *air_temperature* and similar (confidence > 0.5) weather phenomenon types are returned. Click on the “**Name**” column heading to sort the datasets by Name (alphabetical order).
11. Verify that the matching data sets are the same as the ones returned in Step 7.

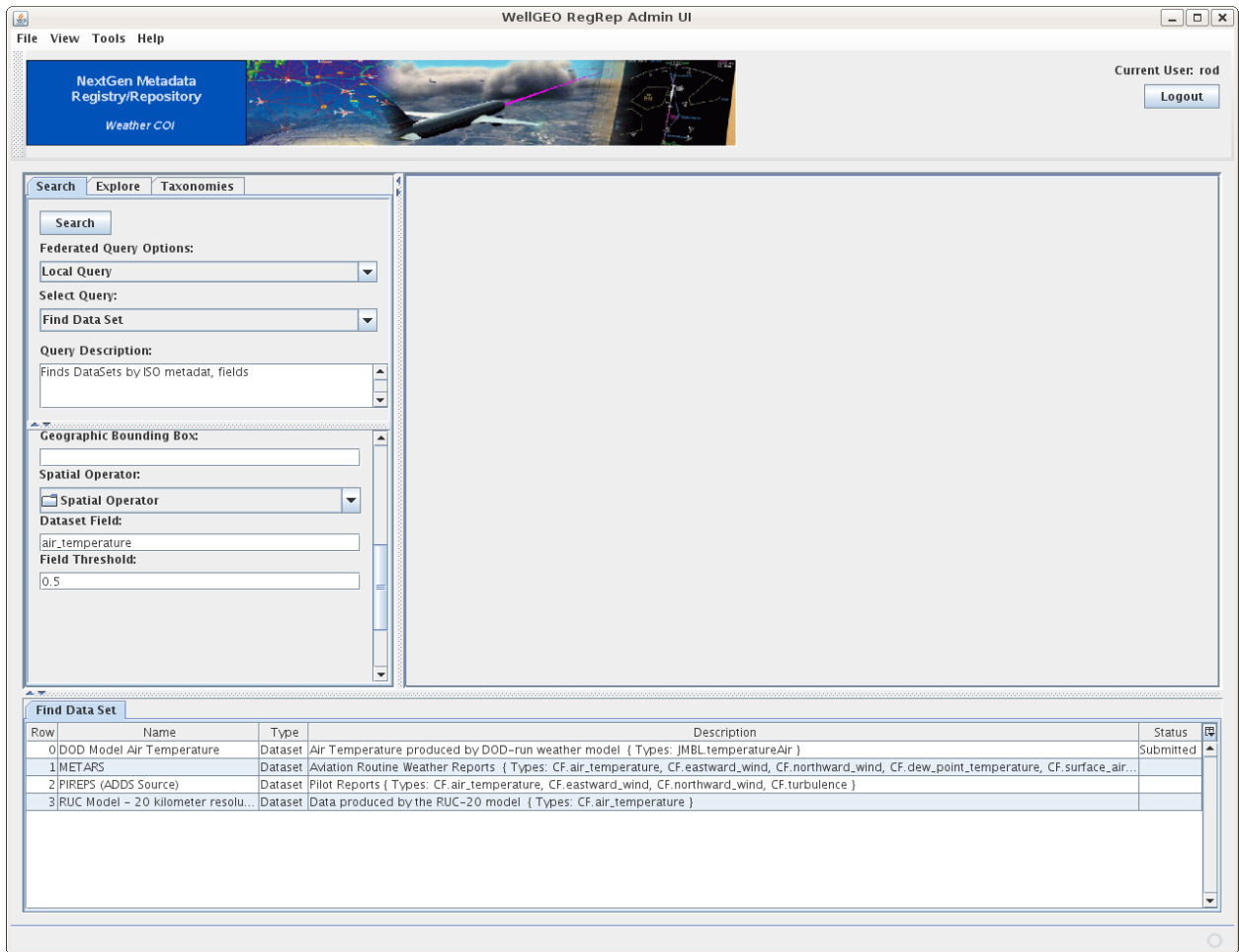


Figure 3.7: Snapshot of the Registry UI showing the results of semantic filtered search, with Data Set Field = air_temperature and Data Set Threshold = 0.5.

12. Repeat the **Search for Data Set** (Step 4). Type the following text in the “**Data Set Field**”:
CloudLiquidWater
13. Leave the “**Data Set Threshold**” blank.
14. Click the “**Search**” button. No records are found as this constitutes a search without ontologies.
15. Type the following value in the “**Data Set Threshold**”:
0.5
16. Click the “**Search**” button. A filtered set of all registered datasets with weather phenomenon type of *CloudLiquidWater* and similar (confidence > 0.5) weather phenomenon types are returned.
17. Verify that the following data set is returned:
 - Vertically Integrated Liquid (VIL)

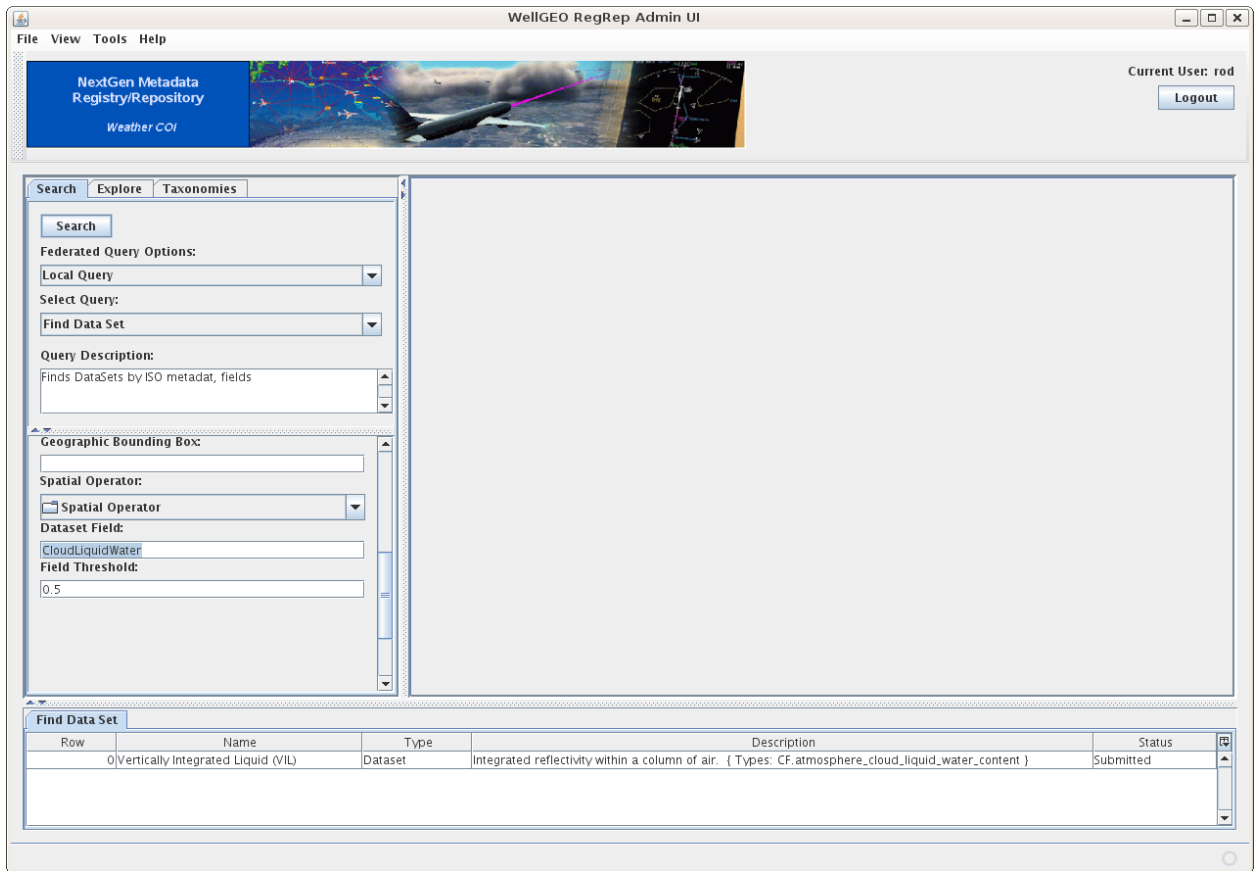


Figure 3.8: Snapshot of the Registry UI showing the results of semantic filtered search, with Data Set Field = CloudLiquidWater and Data Set Threshold = 0.5

18. Type the following value in the “**Data Set Threshold**”:
0.8
19. Click the “**Search**” button. No results should be returned as there are no registered datasets with weather phenomenon type *CloudLiquidWater* or similar (confidence > 0.8) weather phenomenon types.

3.2.6 Dataset Discovery – REST Search Filtered By Dataset Field

This test is identical in function to the earlier test name “Dataset Discovery - Local Search Filtered By Dataset Field”. The only difference is that this test verifies the ability to search the registry using its REST interface using an HTTP GET URL to initiate the search and get results in ebXML RegRep’s ebRIM XML format. The REST interface is suitable for building REST clients to the registry.

The search URL includes the query ID as well as query parameters as URL parameters as shown in the template URL below:

#Template URL for parameterized query invocation

<server base url>/search?queryId=<the query id>&{<param-name>=<param-value>}*&format=<application/xml | application/json>

Applying this to our dataset discovery query filtered by dataset field we have the following URL for the MIT-LL registry:

http://ngenwww2.wx.ll.mit.edu:8080/omar-server-test/rest/search?queryId=urn:ogc:specification:regrep:profile:ISO19139:query:DatasetDiscoveryQuery&field=air_temperature&format=application/xml

1. Click on above URL to open it in a web browser
2. Verify that the matching data sets that are returned include:
 - METARS
 - PIREPS (ADDS Source)
 - RUC Model - 20 kilometer resolution

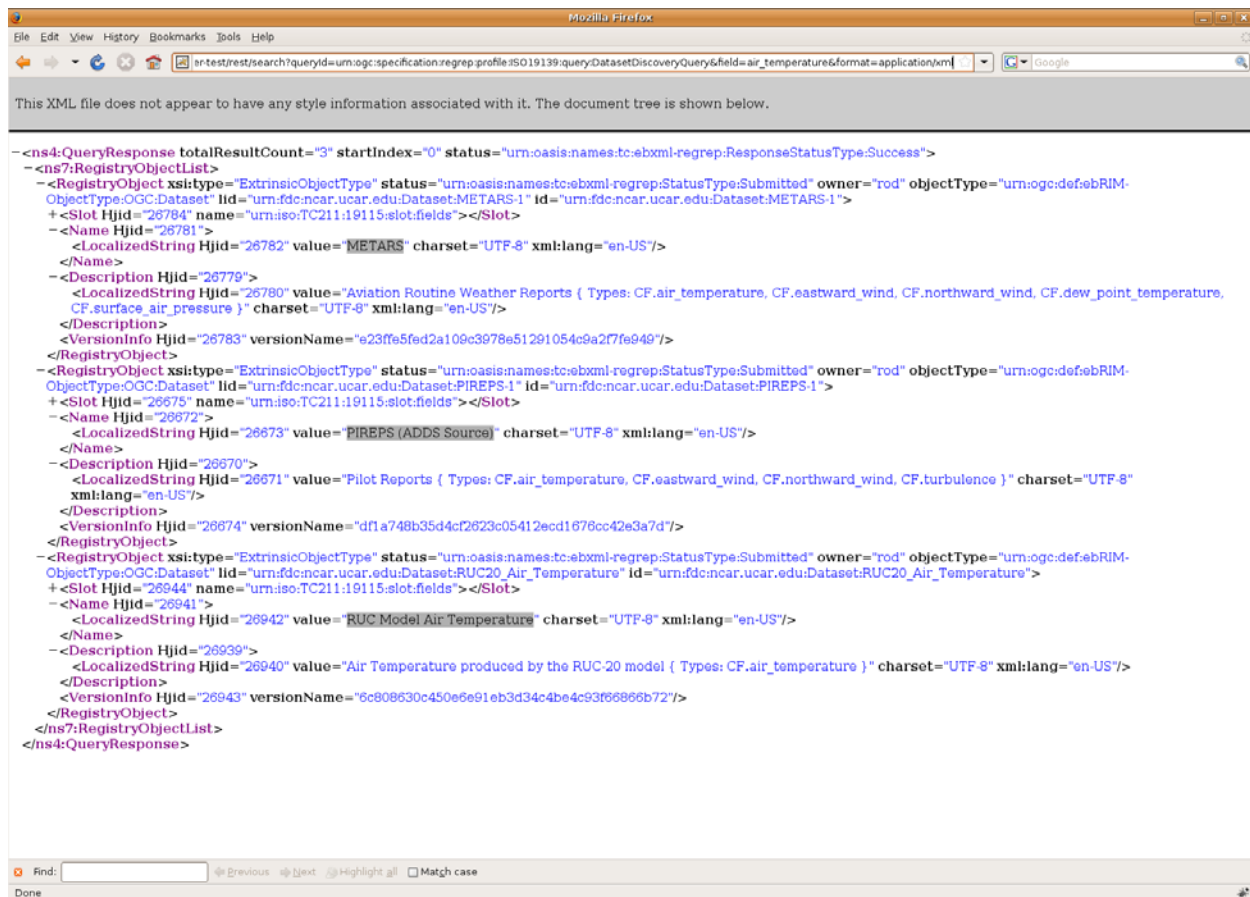


Figure 3.9: Dataset Discovery - REST Search Filtered By Dataset Field = air_temperature

- Now we will change the URL to specify field name “temperatureAir” using the following URL:

<http://ngenwww2.wx.ll.mit.edu:8080/omar-server-test/rest/search?queryId=urn:ogc:specification:regrep:profile:ISO19139:query:DatasetDiscoveryQuery&field=temperatureAir&format=application/xml>

- Click on above URL to open it in a web browser
- Verify that the matching data sets that are returned include:
 - DOD Model Air Temperature

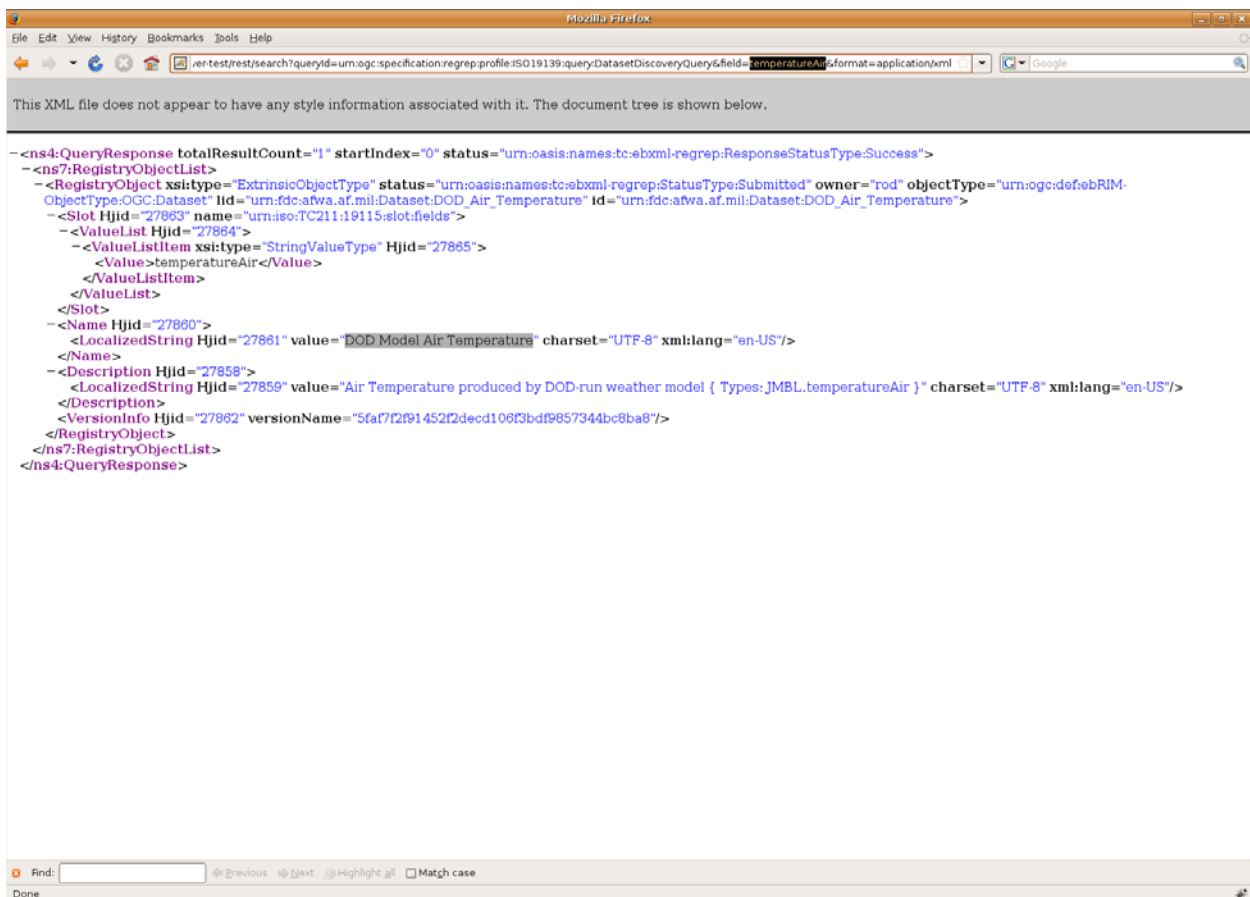


Figure 3.10: Dataset Discovery - REST Search By Dataset Field = temperatureAir

- Now perform the last REST query but specify `format=application/json` instead of `format=application/xml` and verify that the result is returned in JSON format.

<http://ngenwww2.wx.ll.mit.edu:8080/omar-server->

[test/rest/search?queryId=urn:ogc:specification:regrep:profile:ISO19139:query:DatasetDiscoveryQuery&field=temperatureAir&format=application/json](http://mar-server-test/rest/search?queryId=urn:ogc:specification:regrep:profile:ISO19139:query:DatasetDiscoveryQuery&field=temperatureAir&format=application/json)

7. Click on above URL to open it in a web browser
8. Verify that the browser return a JSON format document representing the DOD Model Air Temperature dataset. If you wish to see the document in a pretty formatted then specify above URL in the following web site which offers a JSON formatter:

<http://jsonformat.com/>

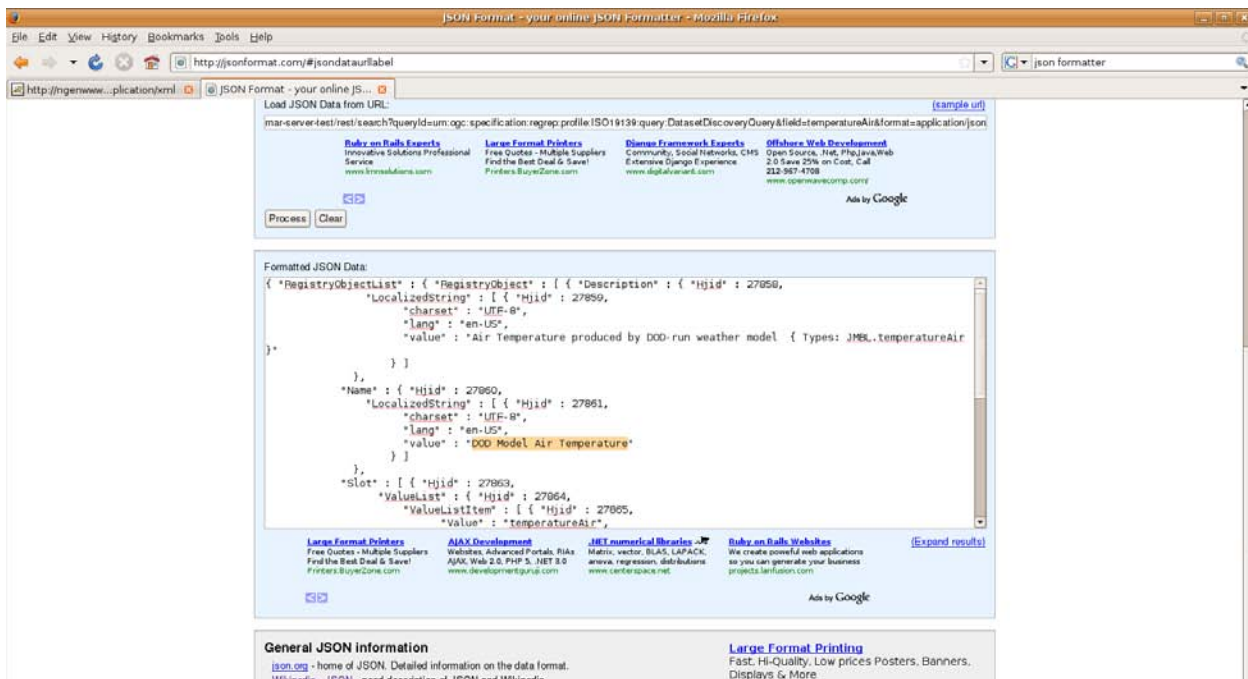


Figure 3.11: Dataset Discovery - REST Search With Results in JSON Format

3.3 Discovery of Datasets by Weather Cube Domain Classification

The weather cube is partitioned into multiple domains and sub-domains. One of the important sub-domains is the Single Authoritative Source (SAS), referred to often in the high-level weather cube CONOPS documents. This test demonstrates browsing for data sets by weather domain capabilities of the Registry/Repository. In particular, it demonstrates:

1. Viewing an existing taxonomy using the Registry Admin UI.
2. Unfiltered Search: Discovery of all registered data sets irrespective of weather cube domain.
3. Filtered Search: Discovery of all registered data sets for a particular sub-domain.

3.3.1 Viewing an Existing Taxonomy

Domains can be organized as a taxonomy. This test performs the viewing of a pre-loaded taxonomy using the Registry Admin UI.

1. Launch the Registry Admin UI **version 4.4** as described in section 3.1.4. By default it will connect with the MIT-LL registry.
2. Click on the “**Taxonomies**” Tab.
3. Scroll down to the “**FAA/NOAA/DOD Data Cube Domain Taxonomy**”
4. Expand the “**Taxonomy**” tree node by clicking on the handle.
5. Verify that the structure of the taxonomy matches the test taxonomy that has been defined and stored in the registry.
 - DataCube
 - Restricted
 - Restricted-Government
 - Restricted-Commercial
 - Unrestricted
 - Regulatory
 - Regulatory-Government
 - Regulatory-Commercial
 - SAS
 - Backup
 - PendingPrimary
 - Primary

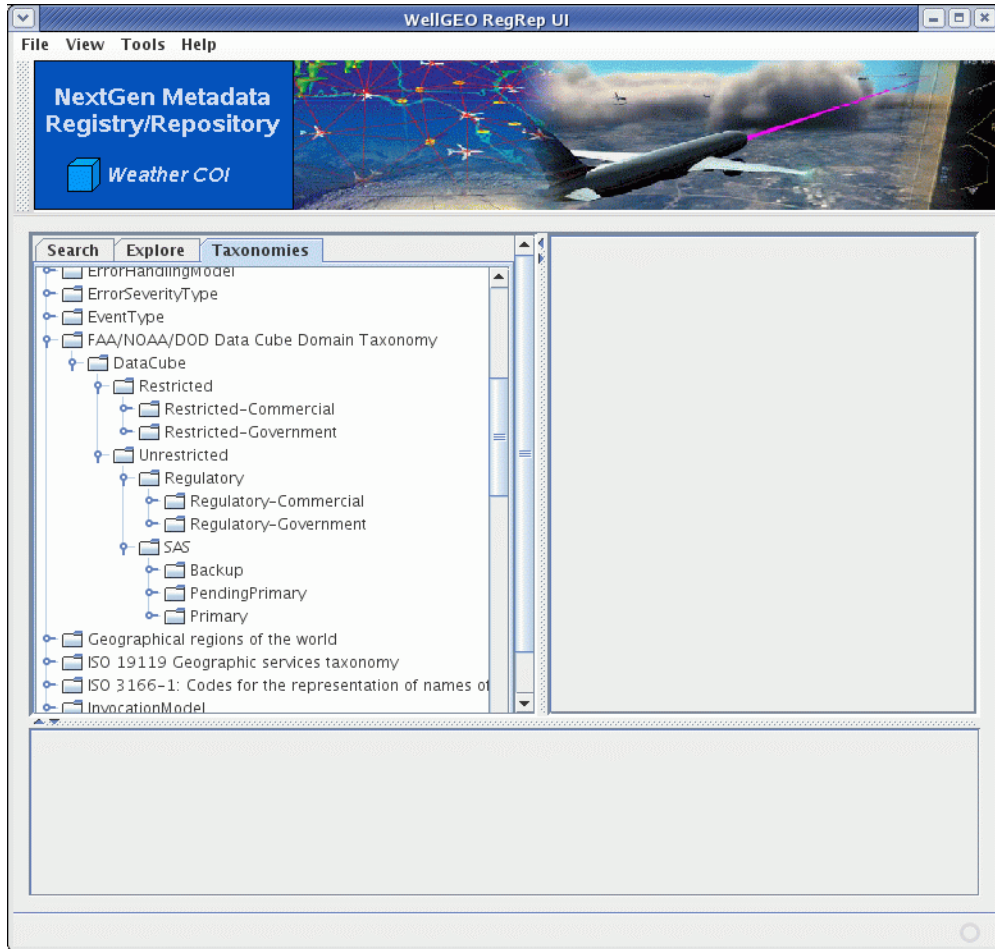


Figure 3.12: Snapshot of the Registry Admin UI showing the test taxonomy

3.3.2 Dataset Discovery - Local Search Filtered By Any Weather Cube Domain

This test performs the discovery of all datasets registered as classified by the data cube domain taxonomy.

1. Launch the Registry Admin UI **version 4.4** as described in section 3.1.4. By default it will connect with the MIT-LL registry.
2. Click on the **“Search”** tab in upper left corner of UI to access the Search Tool panel.
3. Make sure that the Federated Query Options combo box shows **“Local Query”** as selection
4. Select **“Find Data Set”** in the Select Query combo box.
5. Type the following text in the **“Classification”** Field:

DataCube

Note: The '*' is a wildcard character to match zero or more arbitrary characters

6. Click the **“Search”** button. A filtered set of all registered datasets with weather phenomenon classified by the NNEW DataCube taxonomy are returned. Click on the **“Name”** column heading to sort the datasets by name / Name (alphabetical order).

7. Observe the matching data sets that are returned include:
8. All data sets are currently registered as being members of the cube. Verify that all data sets are returned and include the following:
 - Current Icing Potential
 - Current Icing Potential
 - Current Icing Severity
 - DoD Model Air Temperature
 - Echo Tops (Pending SAS)
 - Echo Tops (SAS)
 - Echo Tops (SAS Backup)
 - Flight Category at Surface
 - Geometric Height at Cloud Base
 - Lightning
 - METARS
 - National Convective Weather Forecast Model
 - PIREPS (ADDS Source)
 - RUC Model - 20 kilometer resolution
 - Turbulence
 - Vertically Integrated Liquid (VIL)
 - Visibility at Surface

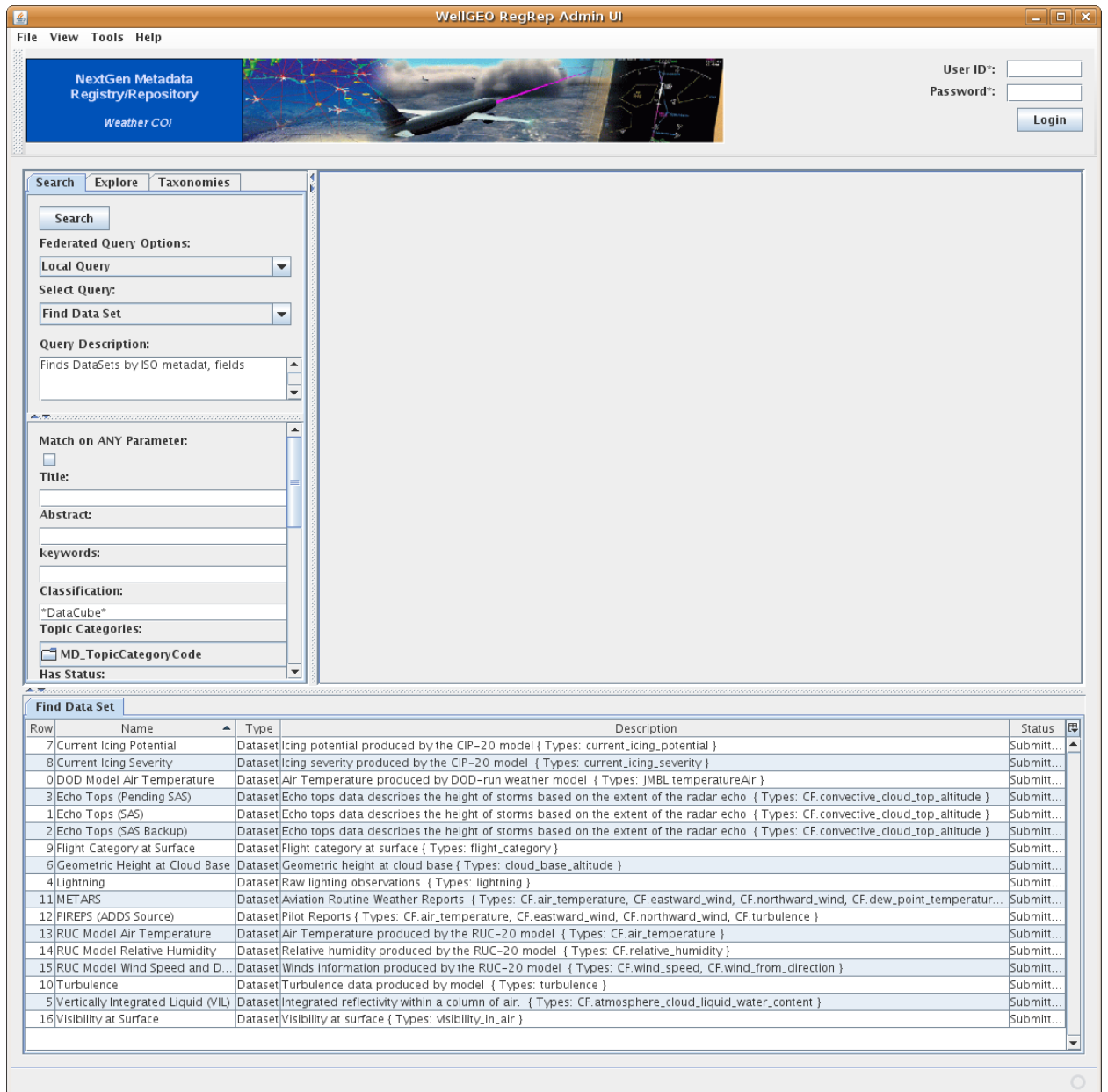


Figure 3.13: Dataset Discovery - Local Search Filtered By *Any* Weather Cube Domain Shows datasets classified by Weather Cube taxonomy from MIT-LL registry

3.3.3 Dataset Discovery - Local Search Filtered By “Unrestricted” Weather Cube Domain

This test performs the discovery of all datasets registered as members of the “Unrestricted” domain.

1. Launch the Registry Admin UI **version 4.4** as described in section 3.1.4. By default it will connect with the MIT-LL registry.
2. Click on the “**Search**” tab in upper left corner of UI to access the Search Tool panel.
3. Make sure that the “**Federated Query Options**” combo box shows “**Local Query**” as selection

4. Select “**Find Data Set**” in the Select Query combo box.
5. Type the following text in the “**Classification**” Field:

DataCube/Unrestricted
6. Click the “**Search**” button.
7. Verify that all data sets in MIT-LL registry with the exception of “**Lightning**” dataset and “**DOD Model Air temperature**” dataset, are registered under the “**Unrestricted**” domain. The lightning data set is classified as “**Restricted**” for commercial reasons.
8. Type the following text in the “**Classification**” field:

DataCube/Unrestricted/SAS
9. Click the “**Search**” button.
10. Verify that the results are the same as that in previous search indicating that all services in “**Unrestricted**” are also in the “**SAS**” domain.
11. Type the following text in the “**Classification**” Field:

DataCube/Unrestricted/SAS/Primary
12. Click the “**Search**” button.
13. Verify that the following Primary, Backup, and PendingPrimary SAS data sets are included in the result set:
 - Current Icing Potential
 - Current Icing Potential
 - Current Icing Severity
 - Echo Tops (SAS)
 - Flight Category at Surface
 - Geometric Height at Cloud Base
 - METARs
 - National Convective Weather Forecast Model
 - PIREPS (ADDS Source)
 - RUC Model - 20 kilometer resolution
 - Turbulence
 - Vertically Integrated Liquid (VIL)
 - Visibility at Surface

3.4 Discovery of Services Instances

A ‘Web Coverage Service’ is any service capable of retrieving data as ISO ‘Coverages,’ which encompass gridded data but also other forms of coverages, such as vertical wind profiles or measurements of weather phenomenon along a trajectory. The OGC Web Coverage Service and JMBL are two examples of services with these capabilities.

This test demonstrates discovery of weather services registered in the Registry/Repository. In particular, it demonstrates:

1. Unfiltered Search: Discovery of all registered service instances.
2. Filtered Search: Discovery of all registered service instances using the ISO 19119 taxonomy.
3. Service Endpoint Retrieval: Retrieve the service endpoint for a specified service instance.

A service instance will be referred to as a service in this document for brevity.

3.4.1 Service Discovery - Unfiltered Local Search

This test performs an unfiltered search for all registered *services* in the MIT-LL Registry.

1. Launch the Registry Admin UI **version 4.4** as described in section 3.1.4. By default it will connect with the MIT-LL registry.
2. Click on the “**Search**” tab in upper left corner of UI to access the Search Tool panel.
3. Make sure that the “**Federated Query Options**” combo box shows “**Local Query**” as selection
4. Select “**Find Service**” in the Select Query combo box.
5. Click the “**Search**” button at the top of the Search Tool panel. An unfiltered list of all the registered services will appear. Click on the “**Name**” column heading to sort the datasets by name / Name (alphabetical order).
6. Verify that the list of services returned includes the following:
 - DOD_JMBLService-01
 - MIT_WebCoverageService-01
 - MIT_WebCoverageService-02
 - MIT_WebCoverageService-03
 - MIT_WebFeatureService-01
 - NCAR_WebCoverageService-01
 - NCAR_WebFeatureService-01
 - NCAR_WebFeatureService-01

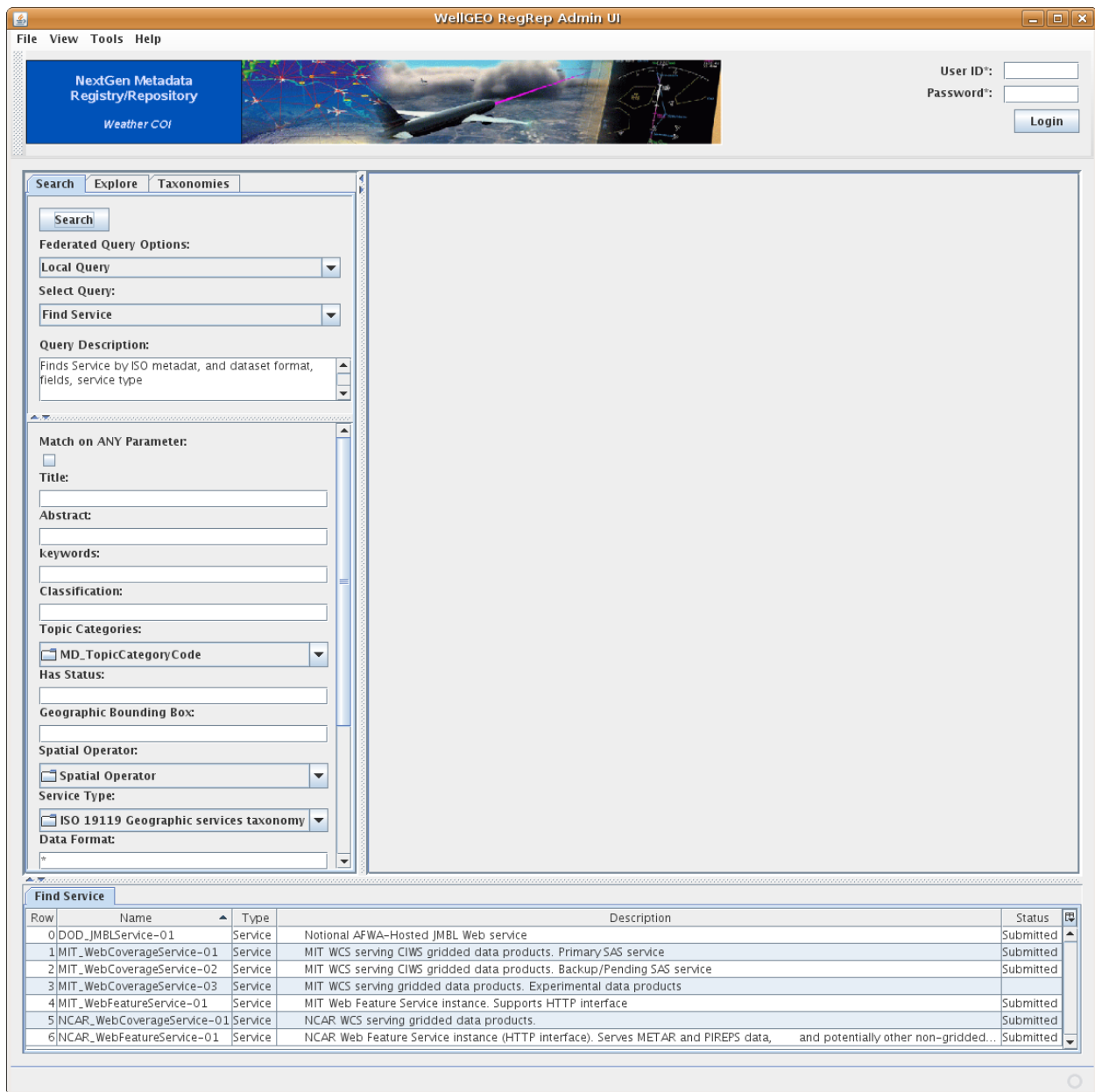


Figure 3.14: Service Discovery - Unfiltered Local Search: matches all service in MIT_LL registry

3.4.2 Service Discovery - Unfiltered Federated Search

This test performs an unfiltered search for all registered *services* across all registries in the NNEW Federation 1 which currently includes MIT-LL and NWS registries.

1. Launch the Registry Admin UI **version 4.5-SNAPSHOT** as described in section 3.1.5.

- Repeat the previous test “Service Discovery - Unfiltered Local Search” with one change; Make sure that in step 3, the Federated Query Options combo box shows “**NNEW Federation 1**” as selection.
- Verify that the list of services returned includes services from both the MIT-LL and NWS registries as indicated by the last column labeled “**Registry**” in search result.

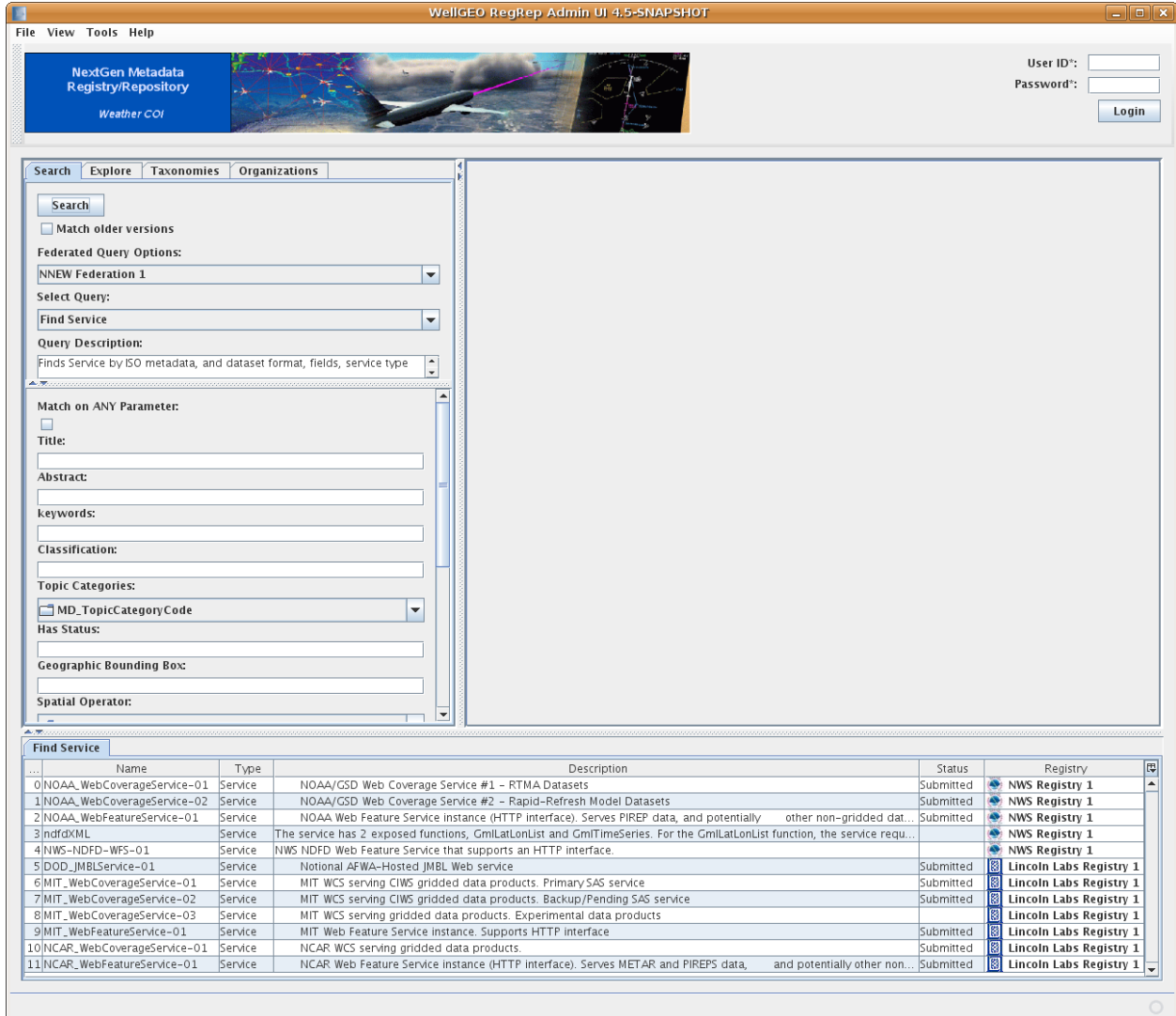


Figure 3.15: Service Discovery - Unfiltered Federated Search
Shows results from MIT-LL and NWS Registries

3.4.3 Service Discovery - Local Search Filtered By Service Type

This test performs the discovery within MIT-LL registry of registered services filtered by service type using the **ISO 19119 Geographic services taxonomy**.

- Launch the Registry Admin UI **version 4.4** as described in section 3.1.4. By default it will connect with the MIT-LL registry.

2. Click on the “**Search**” tab in upper left corner of UI to access the Search Tool panel.
3. Make sure that the “**Federated Query Options**” combo box shows “**Local Query**” as selection
4. Select “**Find Service**” in the Select Query combo box.
5. Select “**Feature Access Service**” in the “**Service Type**” choice box.
6. Click the “**Search**” button. A filtered set of all registered services with service type of “**Feature Access Service**” are returned. Click on the “**Name**” column heading to sort the datasets by name / Name (alphabetical order).
7. Verify that the result set contains the following list of services:
 - DOD_JMBLService-01
 - MIT_WebFeatureService-01
 - NCAR_WebFeatureService-01
 - NCAR_WebFeatureService-01

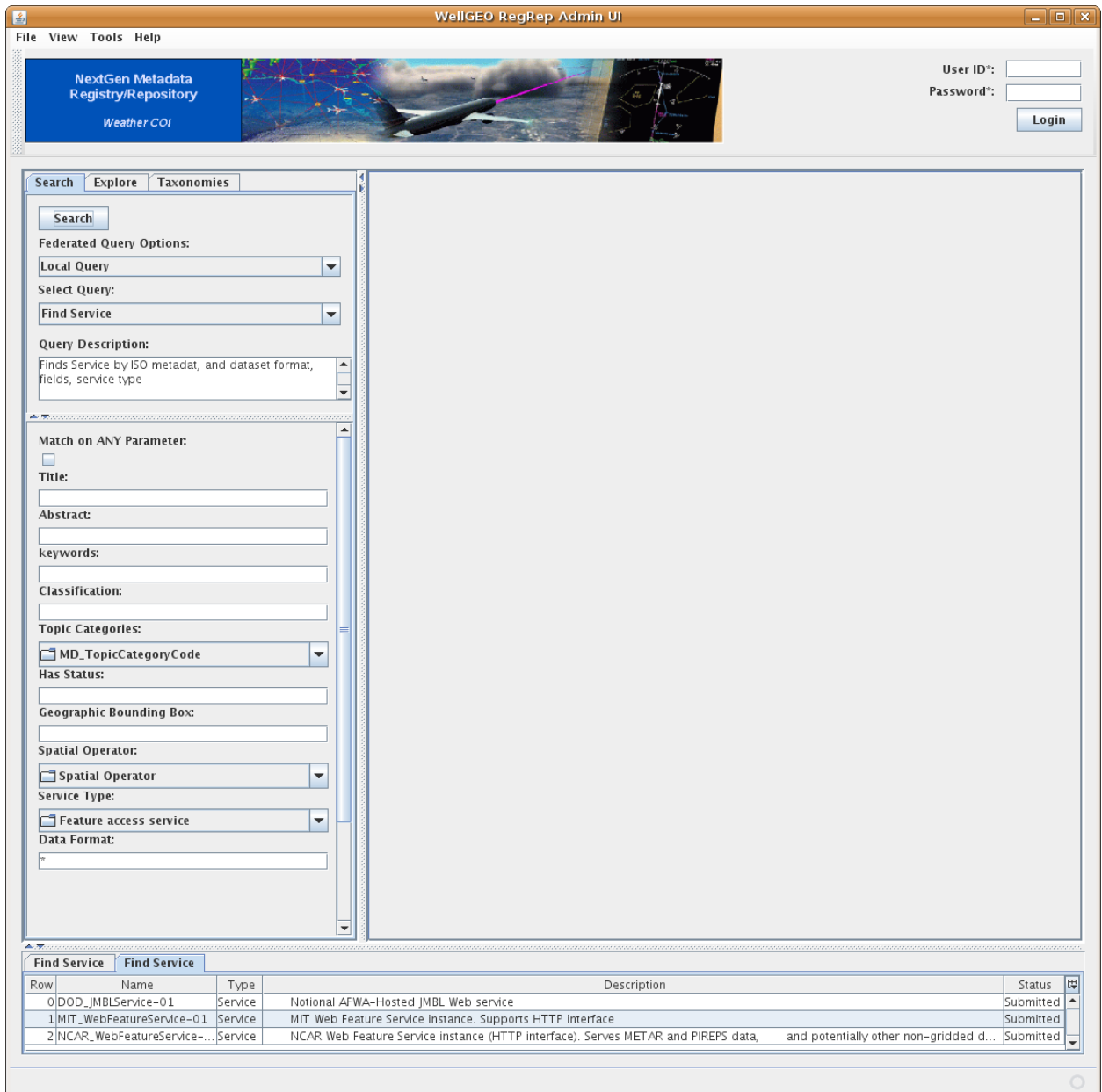


Figure 3.16: Service Discovery - Local Search Filtered By Service Type matches Feature Access Services

8. The Feature Access Service classification used in Step 5 is an abstract classification and does not represent a particular implementation. To refine the query to return only OGC-compliant Feature Access Services (OGC WFS spec), specify the “Web Feature Service (**WFS**) service” sub-type in the “**Service Type**” choice box.
9. Click the “**Search**” button.
10. Click the “**Name**” to sort the services by name.

11. Verify that the result set contains the following OGC WFS feature access services only and not the JMBL service that was matched in previous test:

- MIT_WebFeatureService-01
- NCAR_WebFeatureService-01
- NCAR_WebFeatureService-01

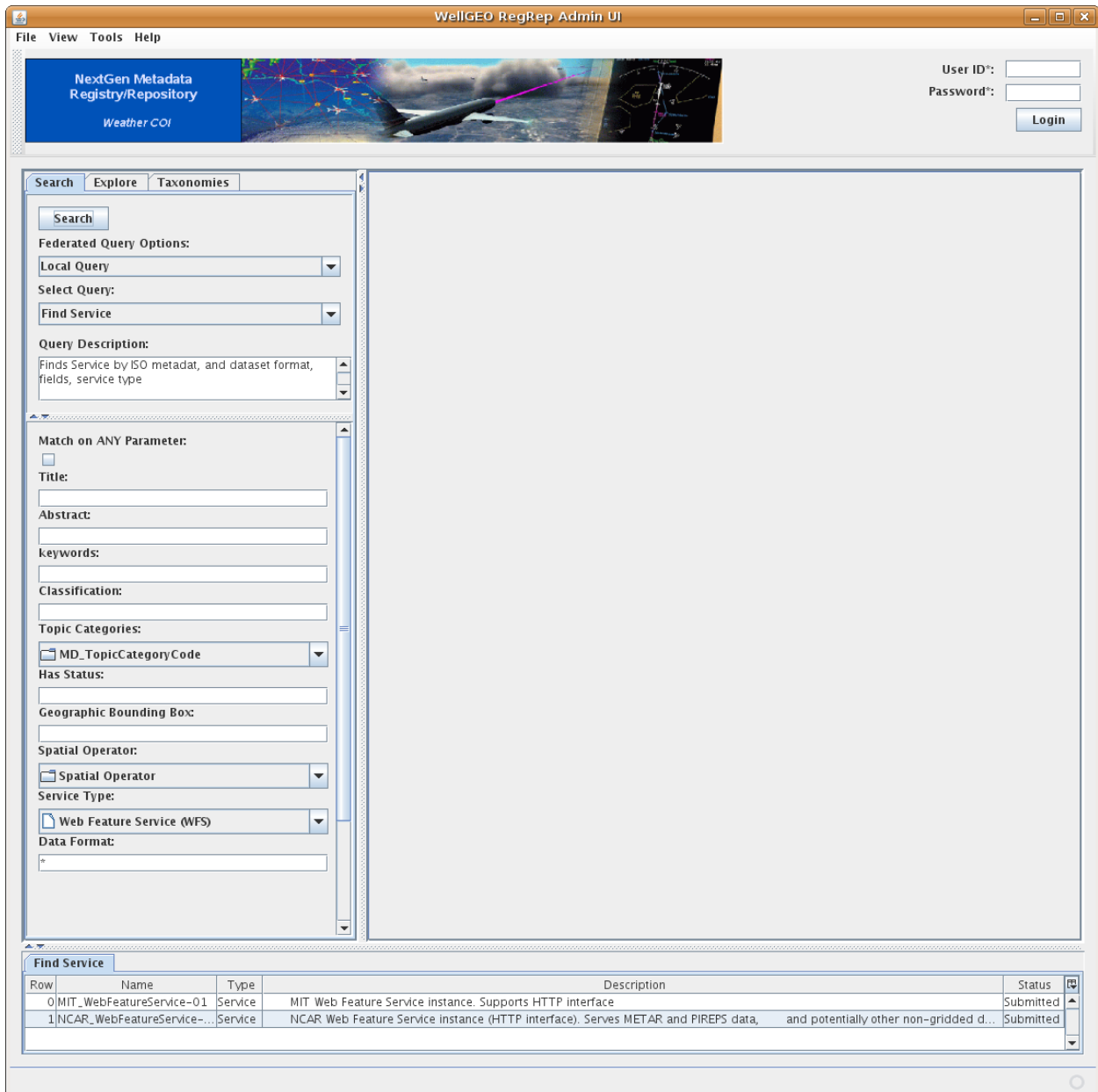
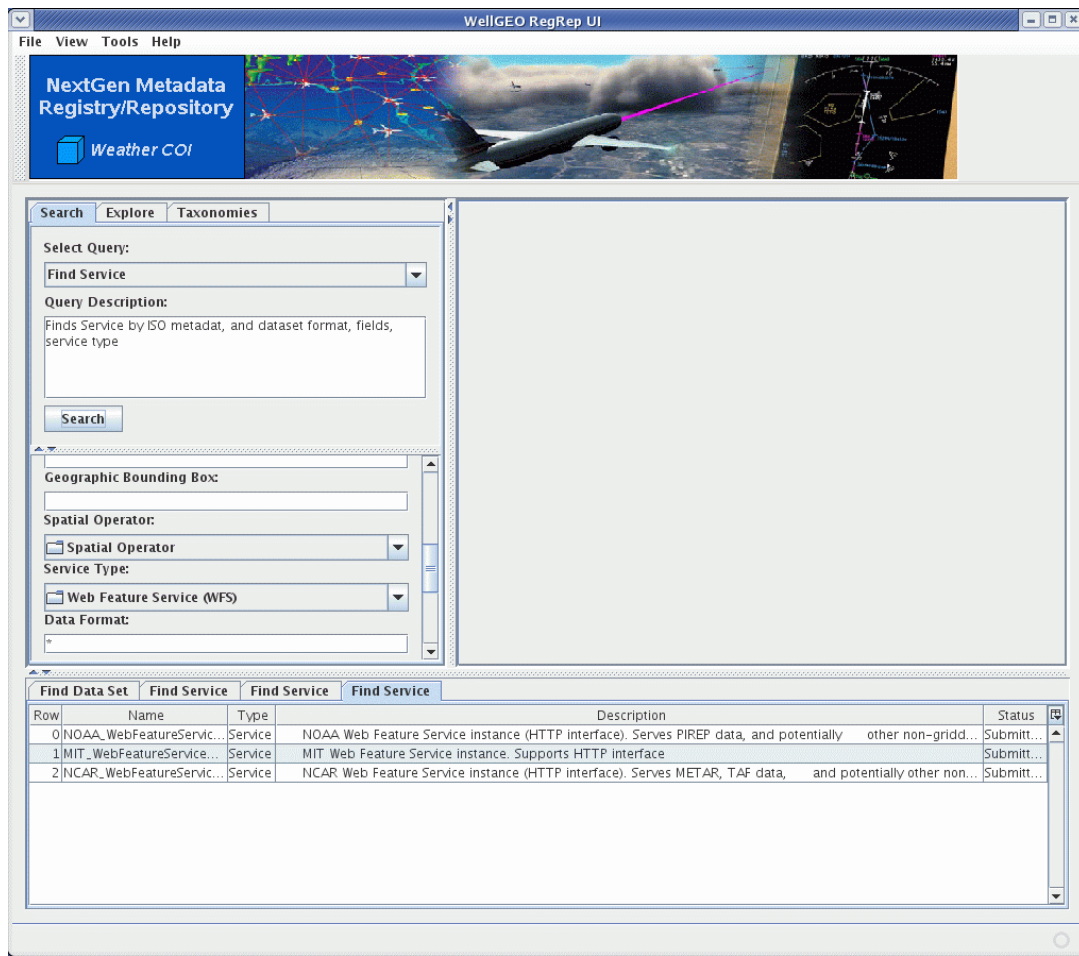


Figure 3.17: Service Discovery - Local Search Filtered By Service Type matches Web Feature Service

12. Repeat Step 5 selecting “**Coverage Access Services**” in the “**Service Type**” choice box.
13. Click the “**Search**” button.
14. Click on the “**Name**” to sort the services by name.
15. Verify that that result set contains the following coverage services. Note that JMBL is capable of acting as both a feature and a coverage access service, so it appears in both coverage and feature access service requests.
 - DOD_JMBLService-01
 - MIT_WebCoverageService-01
 - MIT_WebCoverageService-02
 - NCAR_WebCoverageService-01



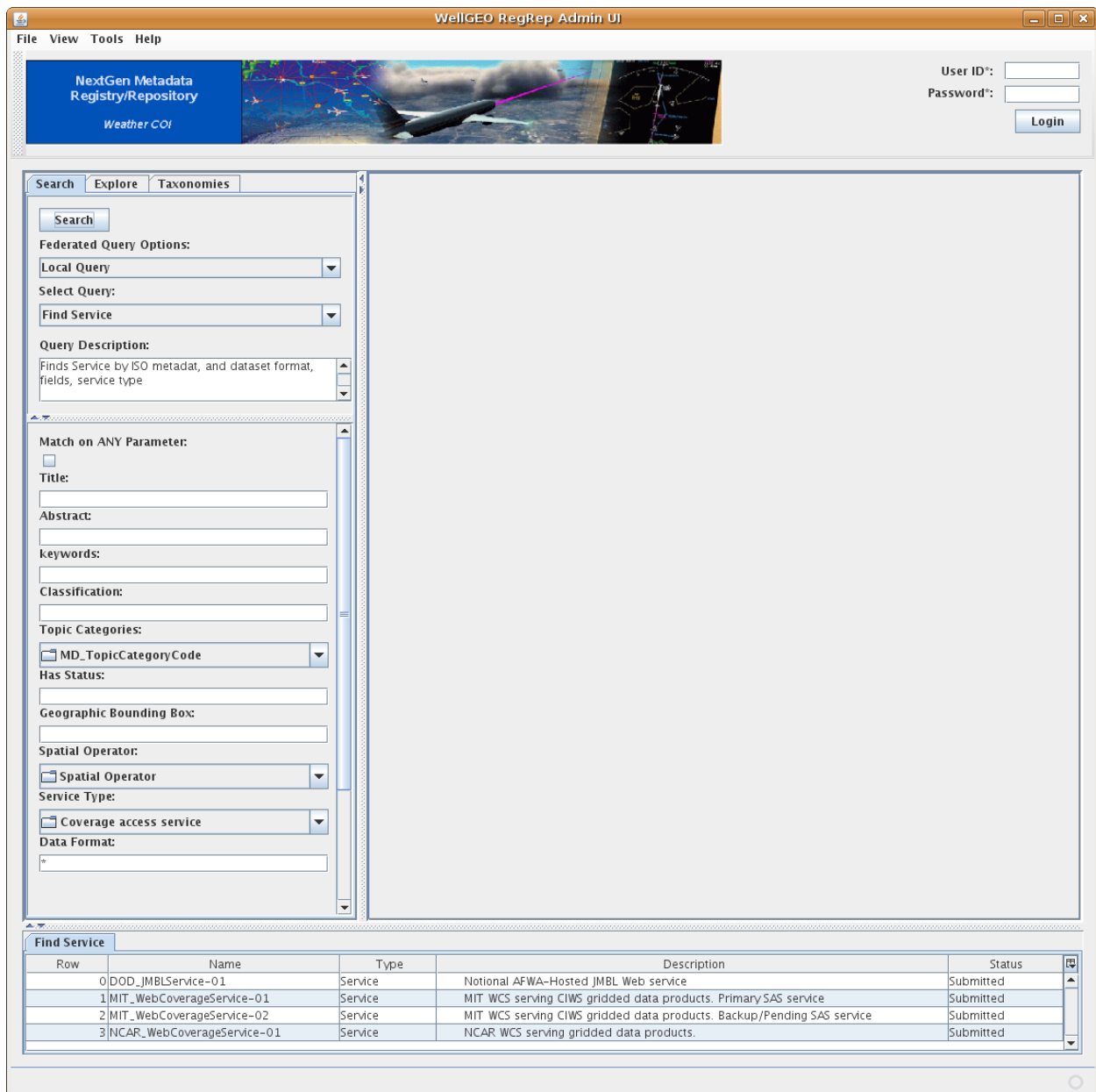


Figure 3.19: Service Discovery - Local Search Filtered By Service Type matches Coverage Access Service

3.4.4 Service Discovery - Federated Search Filtered By Service Type

This test performs the discovery within all registries in **NNEW Fedetaion1** of registered services filtered by service type using the **ISO 19119 Geographic services taxonomy**.

1. Launch the Registry Admin UI **version 4.5-SNAPSHOT** as described in section 3.1.5.
2. Click on the “**Search**” tab in upper left corner of UI to access the Search Tool panel.

3. Make sure that the “**Federated Query Options**” combo box shows “**NNEW Federation 1**” as selection
4. Select “**Find Service**” in the Select Query combo box.
5. Select “**Feature Access Service**” in the “**Service Type**” choice box.
6. Click the “**Search**” button. A filtered set of all registered services with service type of “**Feature Access Service**” are returned.
7. Verify that the list of services returned includes all Feature Access services from both the MIT-LL and NWS registries as indicated by the last column labeled “**Registry**” in search result.

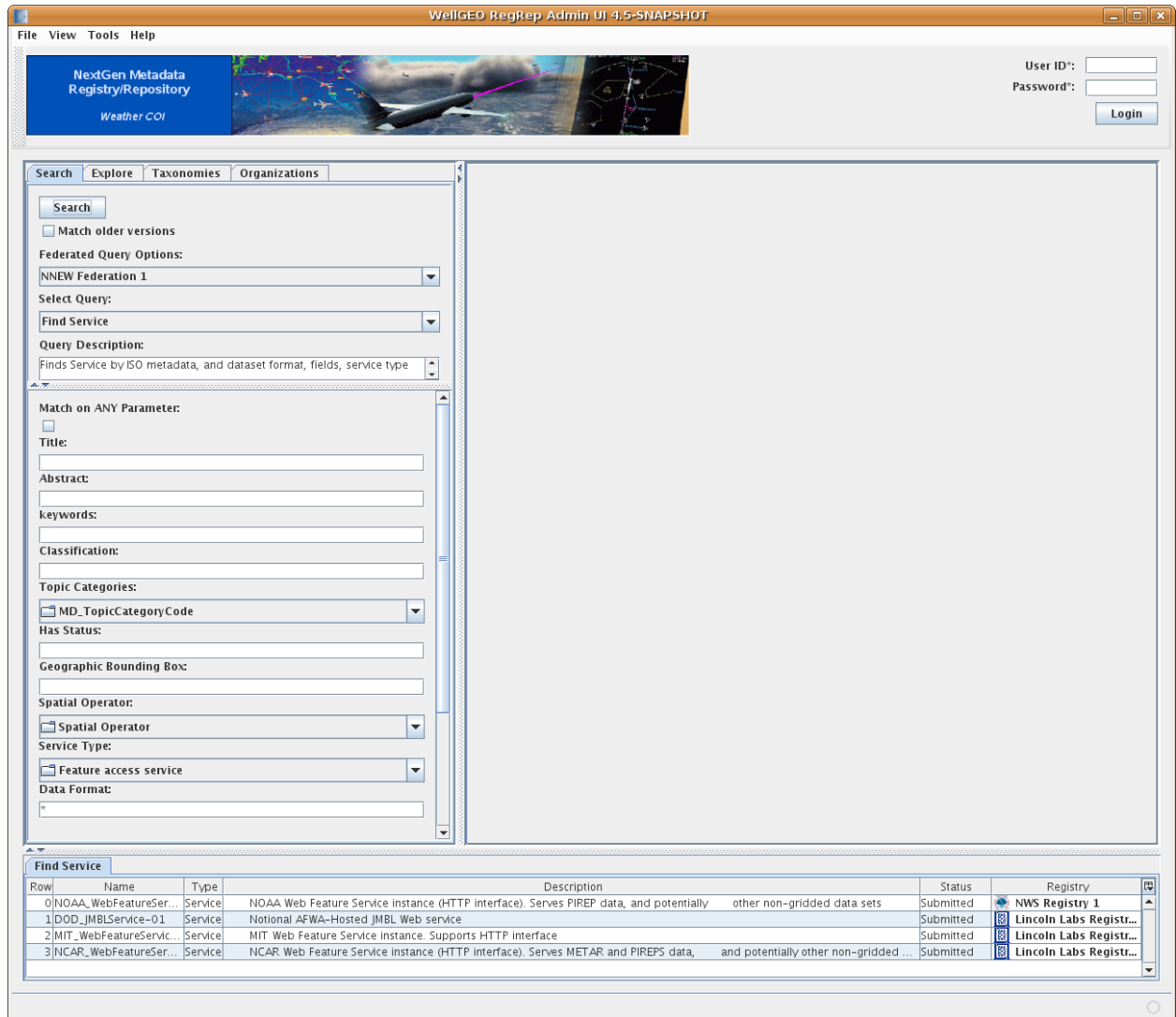


Figure 3.20: Service Discovery - Federated Search Filtered By Service Type matches Feature Access Services in entire Federation

8. Repeat Step 5 selecting “**Coverage Access Services**” in the “**Service Type**” choice box.
9. Click the “**Search**” button.

10. Click on the “Name” to sort the services by name.
11. Verify that the list of services returned includes all **Coverage Access services** from both the MIT-LL and NWS registries as indicated by the last column labeled “Registry” in search result.

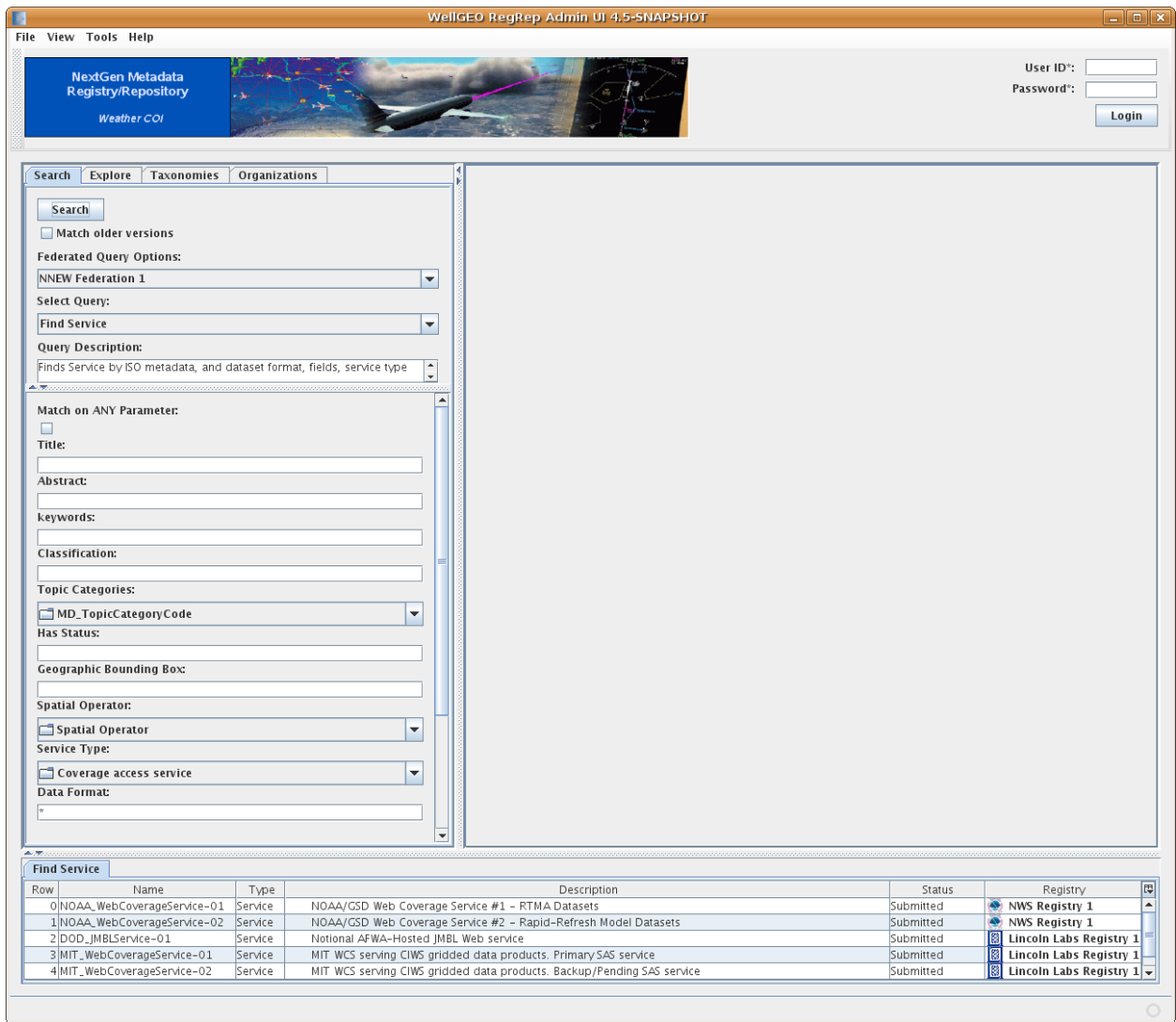


Figure 3.22: Service Discovery - Federated Search Filtered By Service Type Matches Coverage Access Service

3.4.5 Service Endpoint Retrieval

This test retrieves the service endpoint for a selected service.

1. Query for services using the test “Service Discovery - Unfiltered Local Search” from Section 3.4.1.
2. Left-click on the **MIT_WebCoverageService-01** service endpoint in the result set pane to select it. Right-click and select “Open” to open a more detailed service end-point sub-pane.

3. View the detailed information presented in the top right information pane.
4. Left-click on the service endpoint to select it. Right-click to open a more detailed service endpoint sub-pane to display the service endpoint URL (shown at the bottom of the ServiceEndpointType window).
5. Verify that for the MIT_WebCoverageService-01, the URL is:

<http://ngen-wcsri.wx.ll.mit.edu/nnew/fy09/wcs/soap>

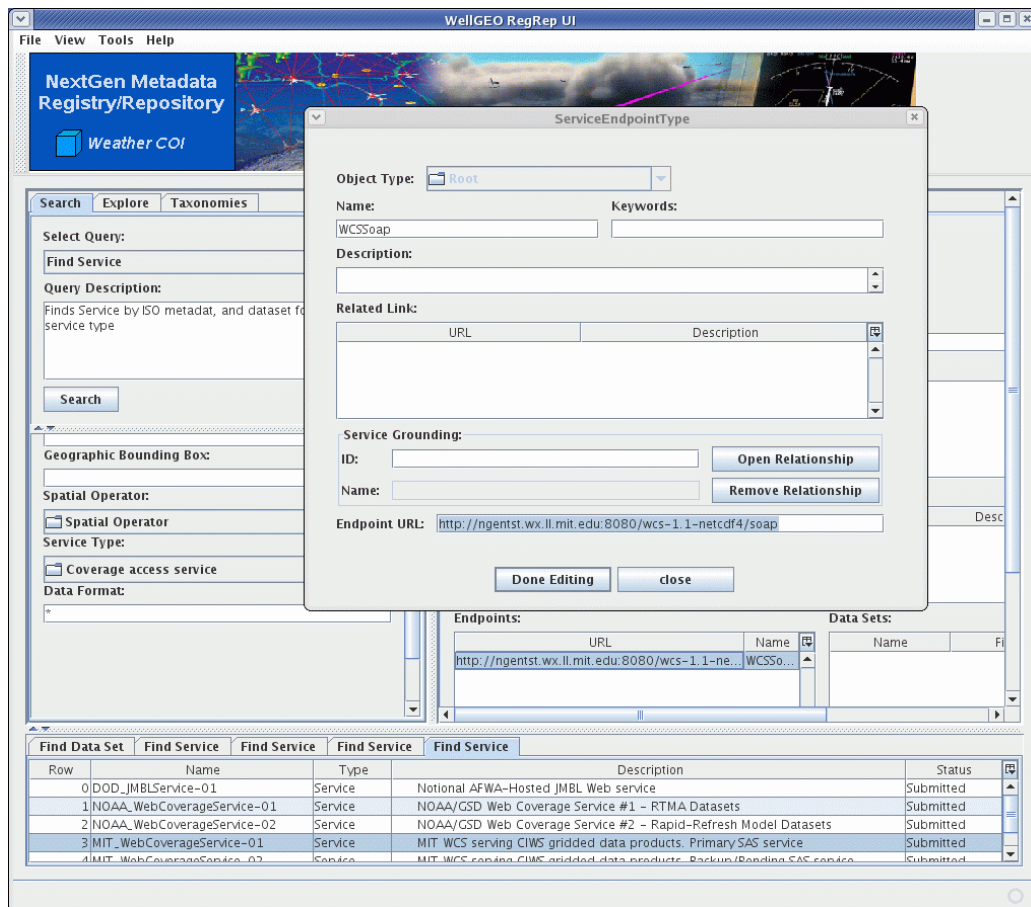
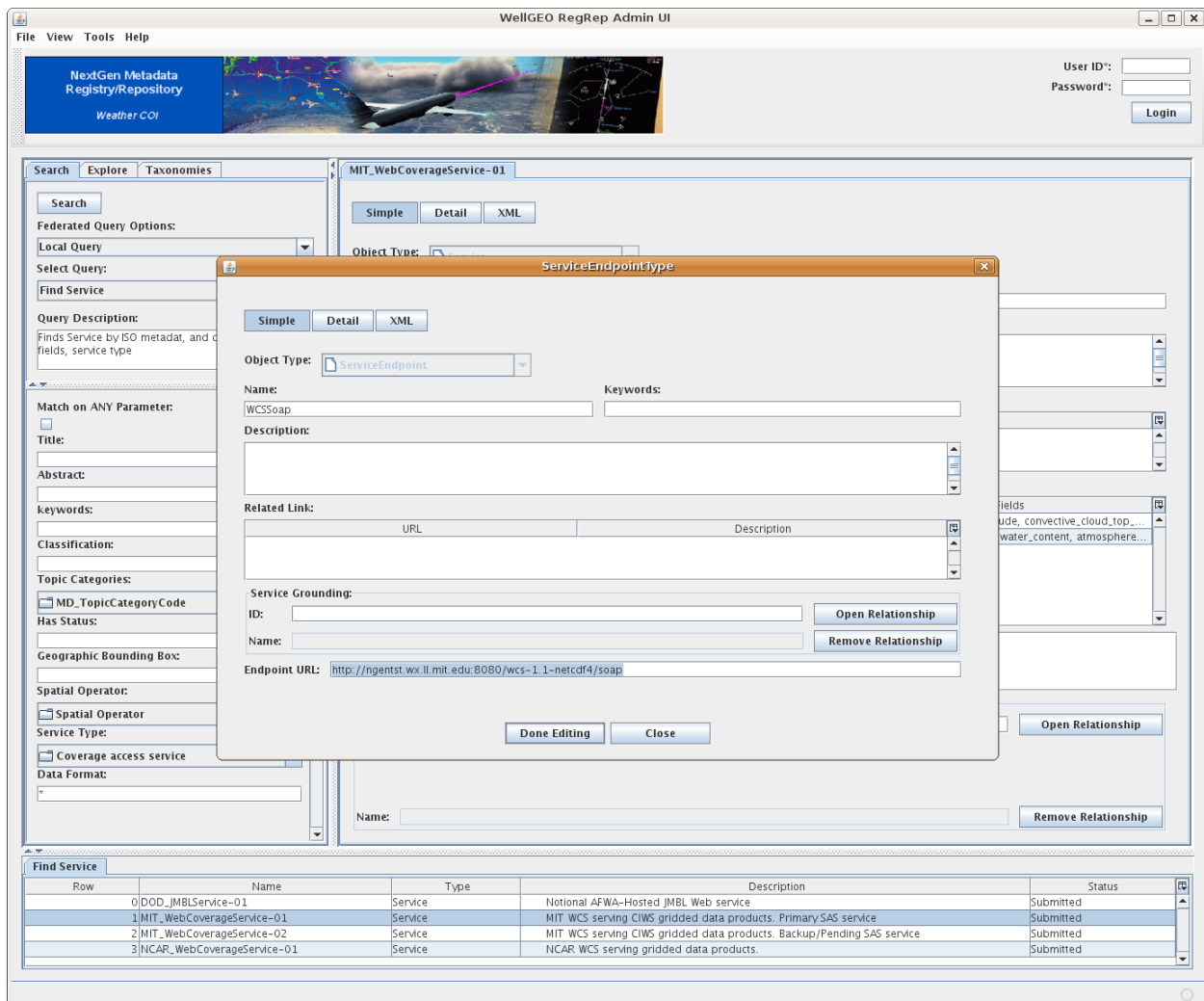


Figure 3.23: Service Endpoint Retrieval



3.4.6 Viewing Datasets Related to a Service

This test verifies that a user can view datasets related to a service. It also illustrates how to view any object that is related to an object using an Association (relationship) as defined by ebRIM in ebXML RegRep.

1. Query for services using the test “Service Discovery - Unfiltered Local Search”.
2. Left-click on the **MIT_WebCoverageService-01** service endpoint in the result set pane to select it. Right-click and select **Open** to open a more detailed service end-point sub-pane.
3. View the information presented in the top right information pane.
4. Click on the “**Detail**” button to show additional details about the service
5. Verify that the Associations table shows to associations of type “**OperatesOn**” with Target Object’s EchoTops and VIL datasets. Note that the 4.5 release displays the values in a more human friendly manner than current 4.4 release.

- Press the Control key (for selecting multiple rows) and the left click the EchoTops and VIL associations in the table so they show selected visual. Now right-click to display context menu. Now select the **“Open Related Objects”** menu.
- Verify that the EchoTops and VIL datasets are displayed in new tabs within the editor panel.

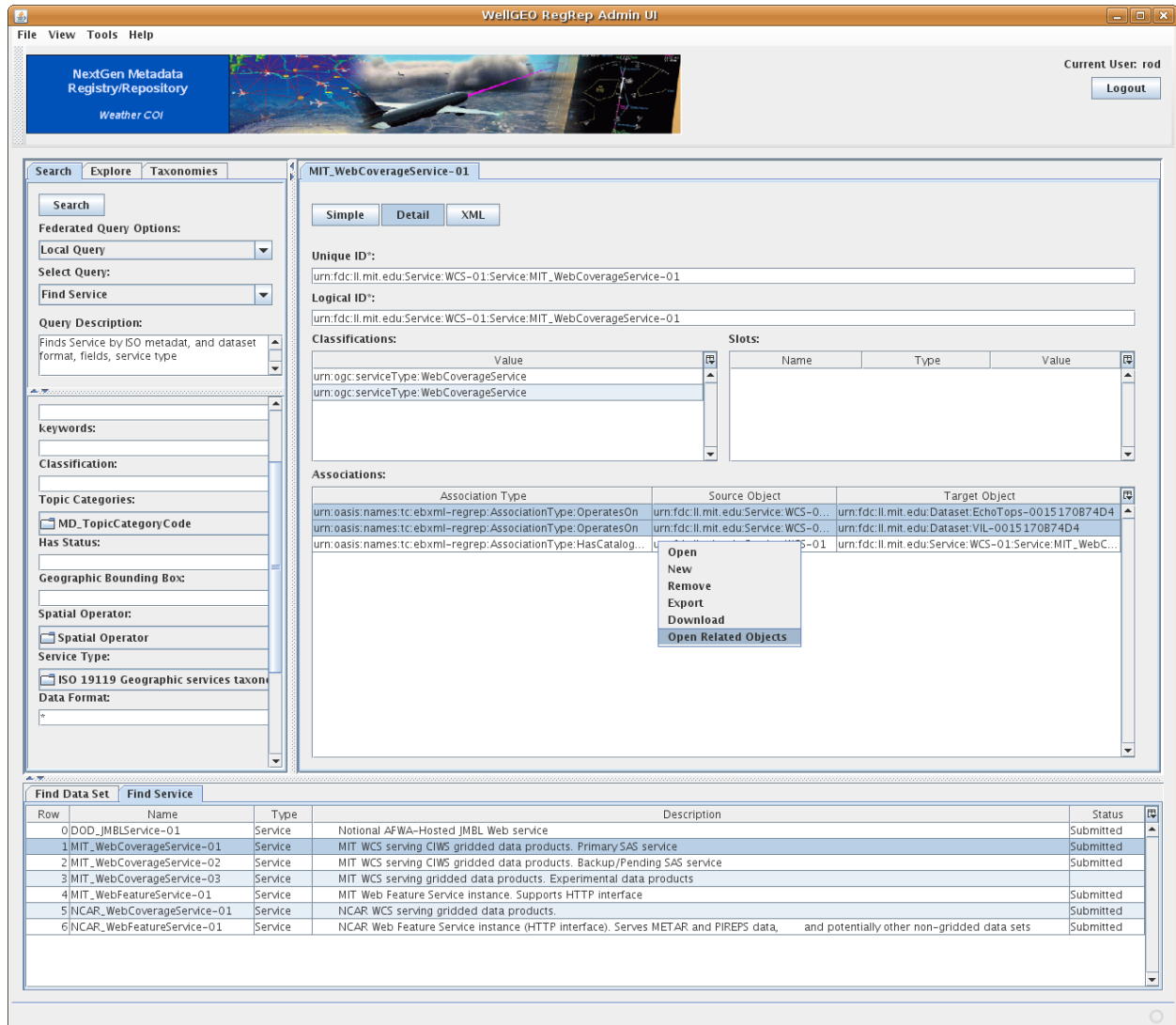


Figure 3.24: Viewing Datasets Related to a Service

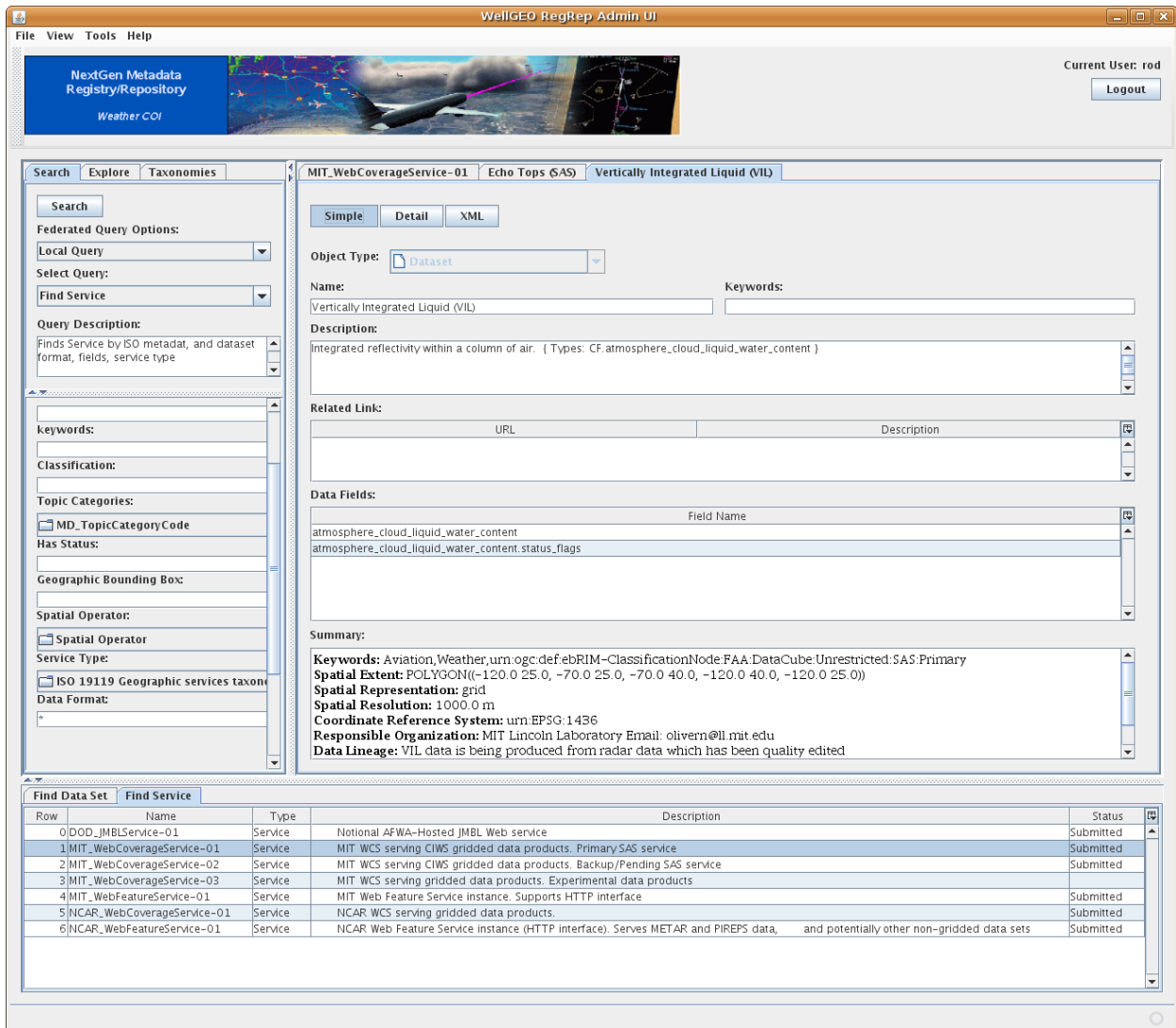


Figure 3.25: Opening Datasets Related to Service

3.5 Creation of an Experimental Weather Cube Taxonomy

The research community is continually refining weather products and generating new weather products. For testing and evaluation purposes, it is often convenient to have available a test version of the weather cube that provides information about the experimental data sets. The registry is capable of storing information about an arbitrary number of weather domains, as they are simply stored within the registry as taxonomies.

An experimental taxonomy is provided with the test data package to test out the loading of custom weather domains. As constructed, it essentially mimics the official weather cube domain, but different unique identifiers are used for all the taxonomy nodes. This is for test purposes - custom weather domains are not necessarily limited to the same structure as the official weather domain taxonomy.

This test demonstrates the loading of a new taxonomy and verifies that the loaded taxonomy is similar to the weather cube domain taxonomy

1. Launch the Registry Admin UI **version 4.4** as described in section 3.1.4. By default it will connect with the MIT-LL registry.
2. Login to the registry using a valid User ID and Password.
3. Select “**Tools → Wizards → Import Files**” to open the file import window.
4. Browse to the \$TEST_DATA_DIR/wxcube-metadata/taxonomies directory provided with the test package.
5. Select the experimental version of the data cube domain taxonomy:
ebrim-classificationScheme-TriAgencyDataCubeExp.xml
6. Click on “**Finish**” to perform the load. After a short pause, an indication of load success is returned.
7. Close the file import window.

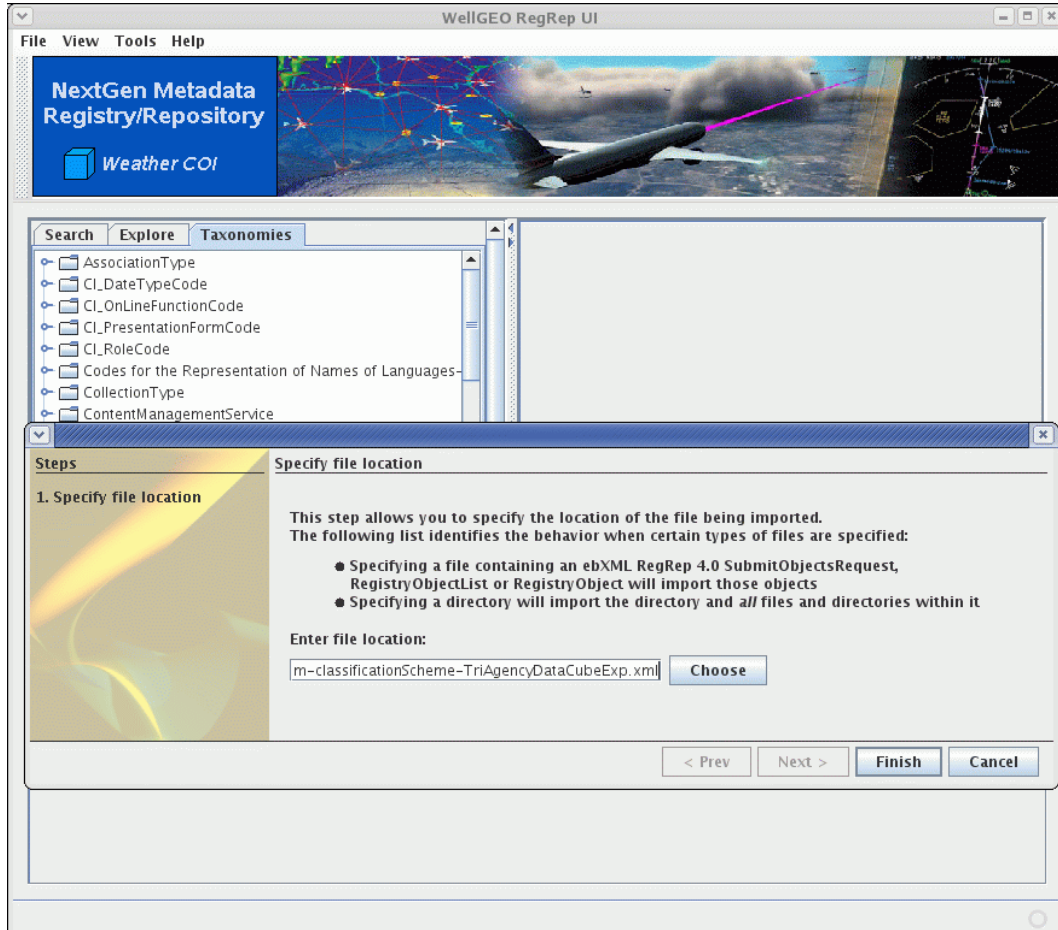
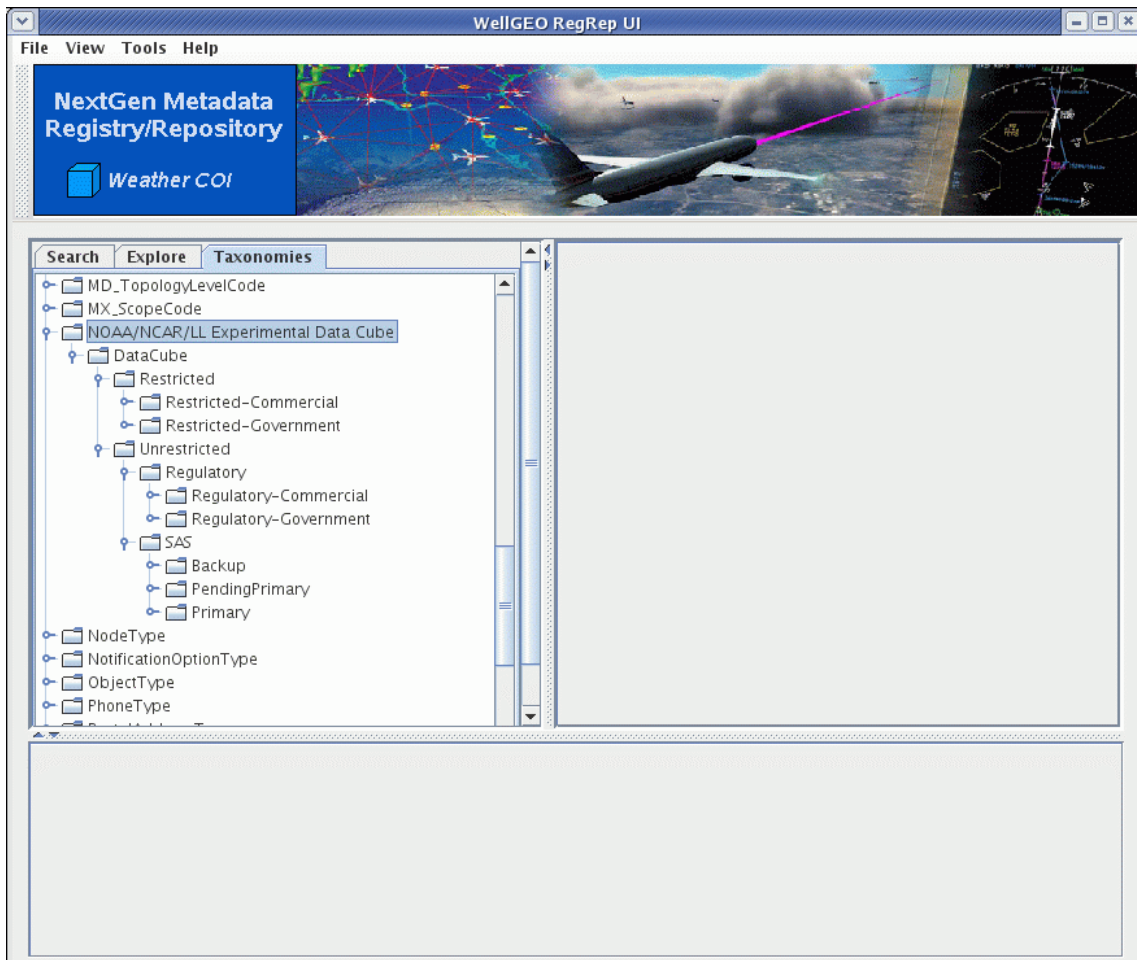


Figure 3.26: Snapshot of the Registry Admin UI displaying the interface for loading a taxonomy.

8. The taxonomy now resides in the registry. However, the Registry Admin UI session isn't automatically aware of it (if the UI is restarted, it will be loaded). In order to refresh the Registry Admin UI without restarting the application, use **View → Reload**. This operation takes approximately 30 seconds, so please be patient.
9. Click the “**Taxonomies**” tab. Navigate to “**NOAA/NCAR/LL Experimental Data Cube Taxonomy**. ”
10. Browse the taxonomy tree and verify that it matches up with the official weather cube taxonomy.
 - DataCube
 - Restricted
 - Restricted-Government
 - Restricted-Commercial
 - Unrestricted
 - Regulatory
 - Regulatory-Government
 - Regulatory-Commercial
 - SAS
 - Backup
 - PendingPrimary
 - Primary



3.6 Publication of an Experimental Data Set and Accompanying Experimental Data Access Service

This test demonstrates the publication of an experimental data set and an accompanying data access service. In order to demonstrate publication of datasets and related services, metadata for an additional (not pre-loaded) experimental dataset and service is included in the test package.

Note that although this test exercises dataset and service publication in an experimental context, the overall actions and results are the same for non-experimental datasets.

3.6.1 Publish Dataset

This test demonstrates the publication of an experimental data set.

1. Launch the Registry Admin UI **version 4.4** as described in section 3.1.4. By default it will connect with the MIT-LL registry.
2. Login in using a valid **User ID** and **Password** using the login controls in upper right of the screen.
3. Open the “**Register Datasets**” wizard using “**Tools**” → “**Wizards**” → “**Register Datasets**” in the menu bar.

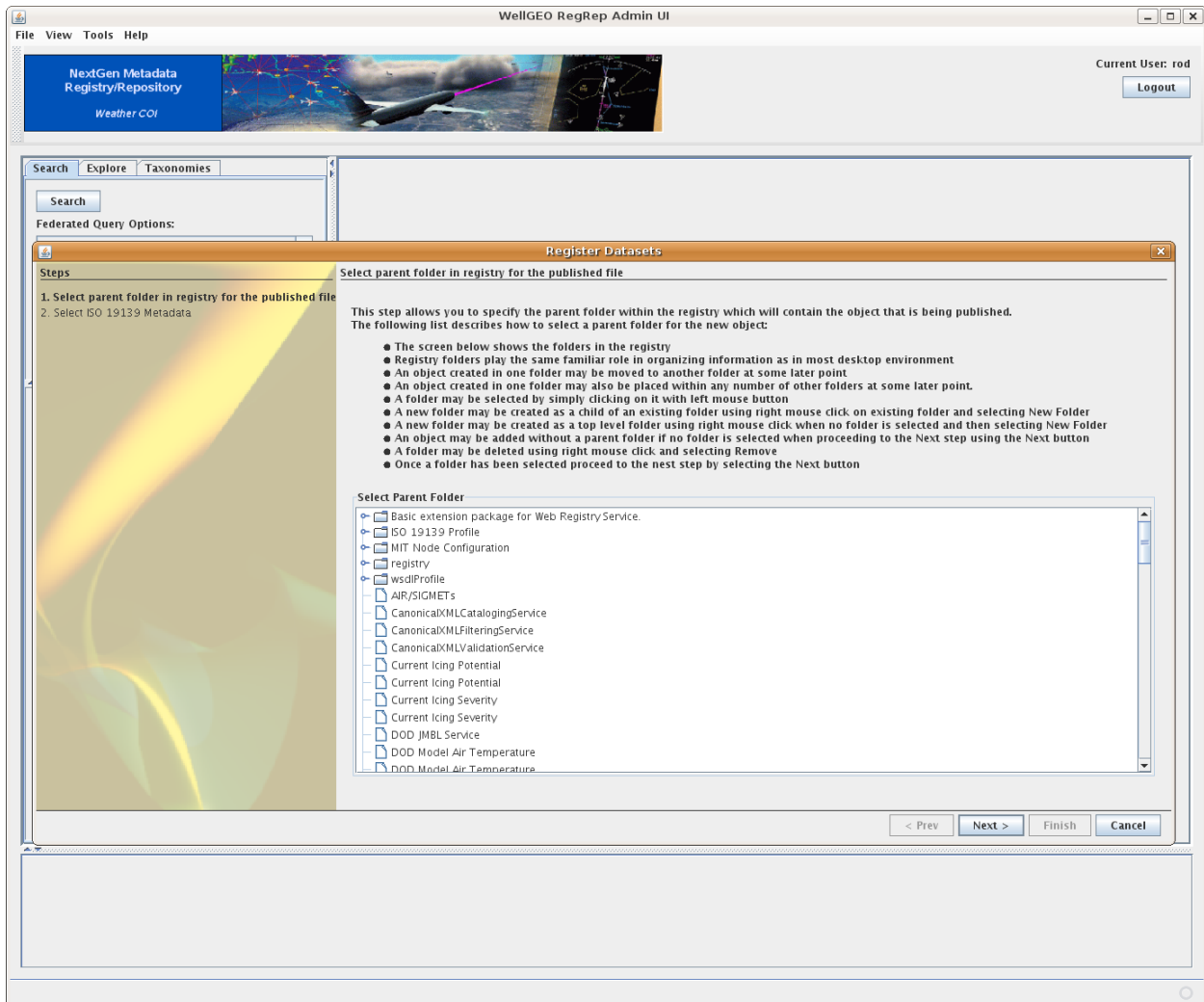


Figure 3.28: Dataset Publish - Select Parent Folder

4. Click on the Next button to skip the optional step 1 (“**Select Parent Folder**”) and go to Wizard step 2 (“**Select ISO 19139 Metadata**”) as shown in steps window at left side of Wizard.
5. Browse to \$TEST_DATA_DIR/wxcube-metadata/datasets/mit-ll, and select the file: iso-metadata-dataset-EchoTops-Exp-01.xml

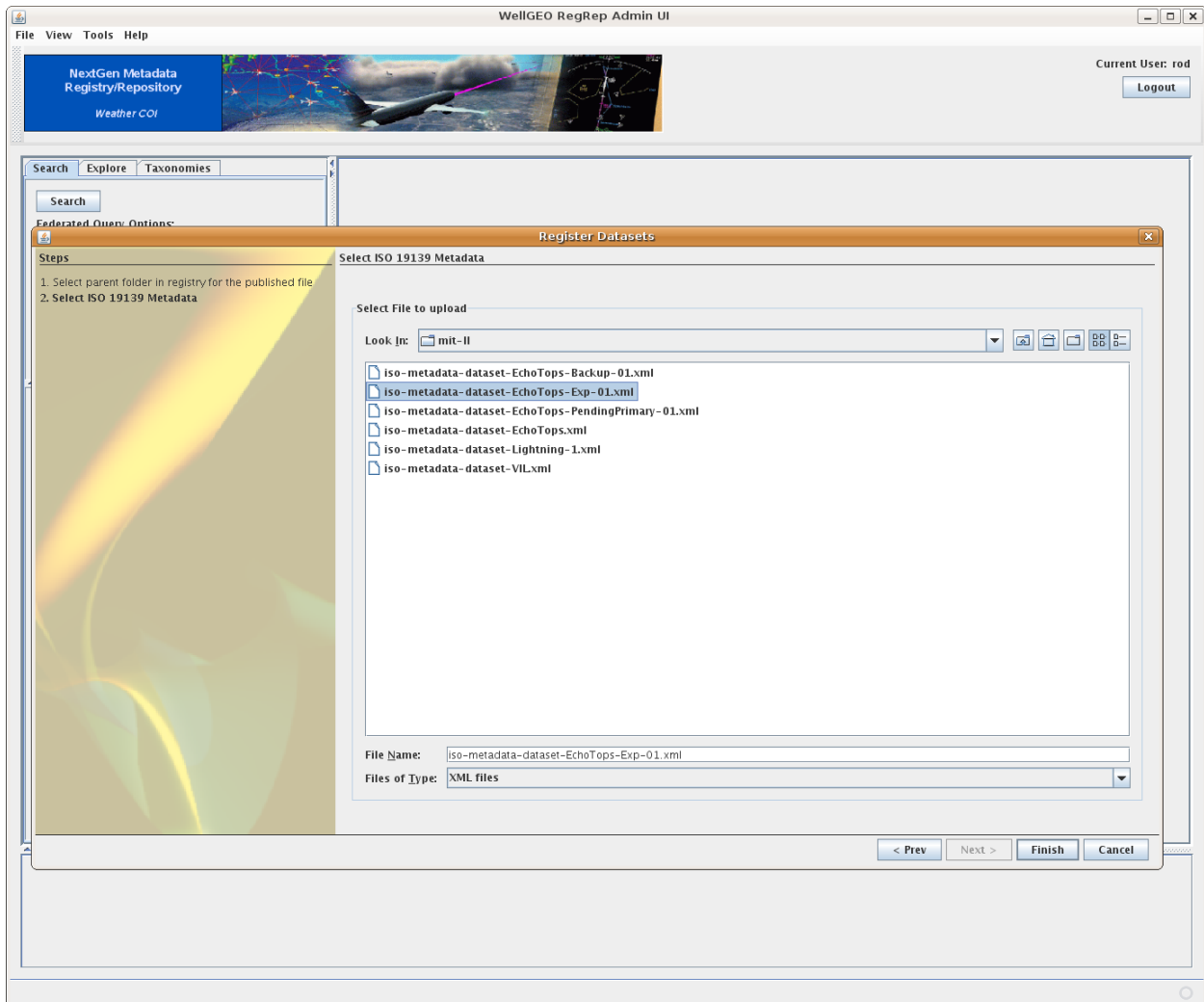


Figure 3.29: Dataset Publish - Select ISO 19139 Metadata File

6. Click on the “**Finish**” button to publish the dataset specified in the ISO 19139 metadata file. An indication of success is returned in the Summary page.
7. Verify that the dataset was successfully published by searching for all datasets in the registry. Added to the former list of datasets should be one with the name Echo Tops (Experimental).

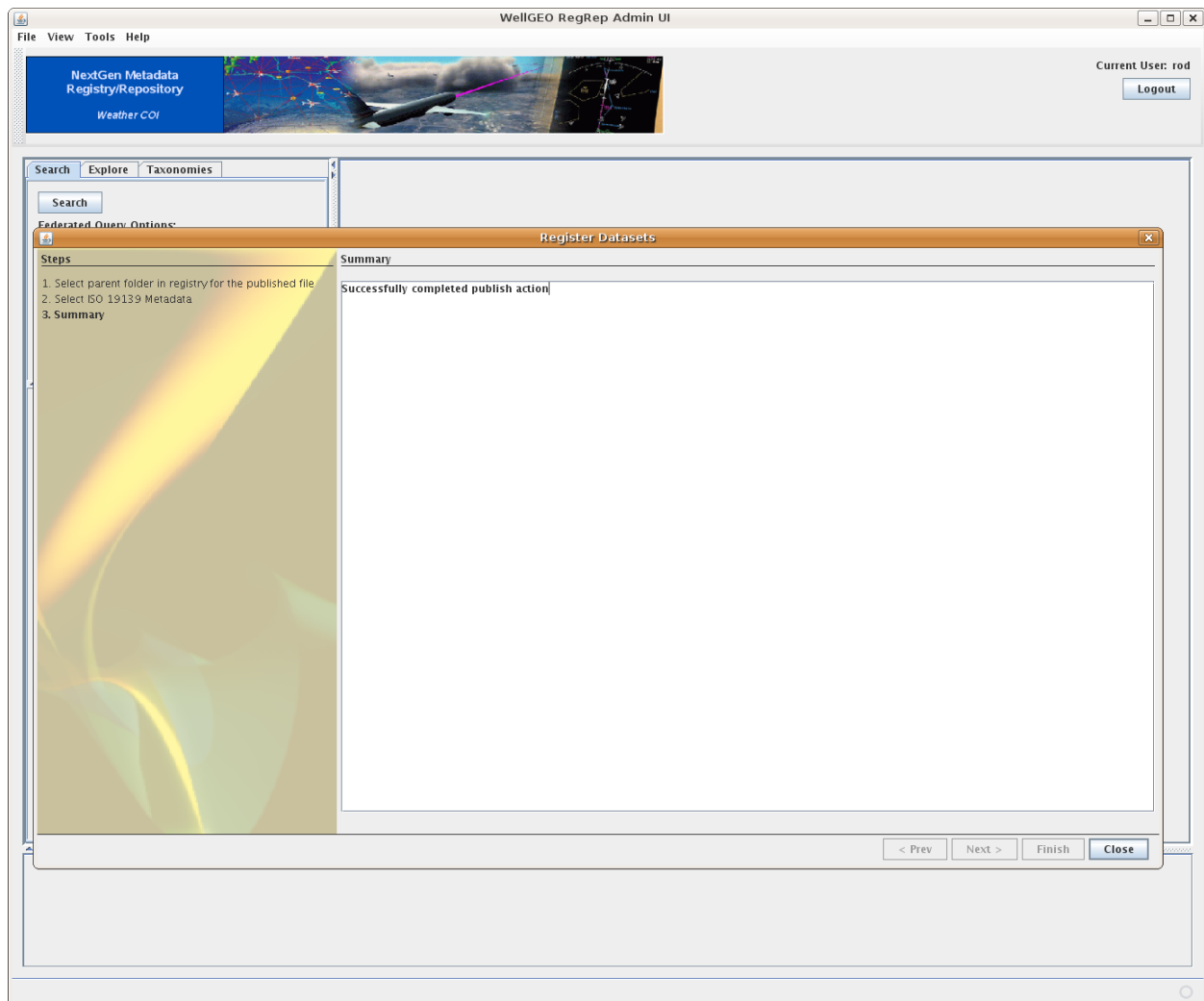


Figure 3.30: Dataset Publish - Finish Screen

3.6.2 Publish Service Instance

This test published a service instance to the registry. Publication of services currently requires that two items be specified, a WSDL file and an accompanying ISO 19139 metadata file. In the Publish Service Instance wizard, the WSDL file is specified first, followed by the ISO 19139 file.

1. Launch the Registry Admin UI **version 4.4** as described in section 3.1.4. By default it will connect with the MIT-LL registry.
2. Login in using a valid **User ID** and **Password** using the login controls in upper right of the screen.

3. Open the “**Register Service Instance**” wizard using “**Tools**” → “**Wizards**” → “**Register Service Instance**” in menu bar.

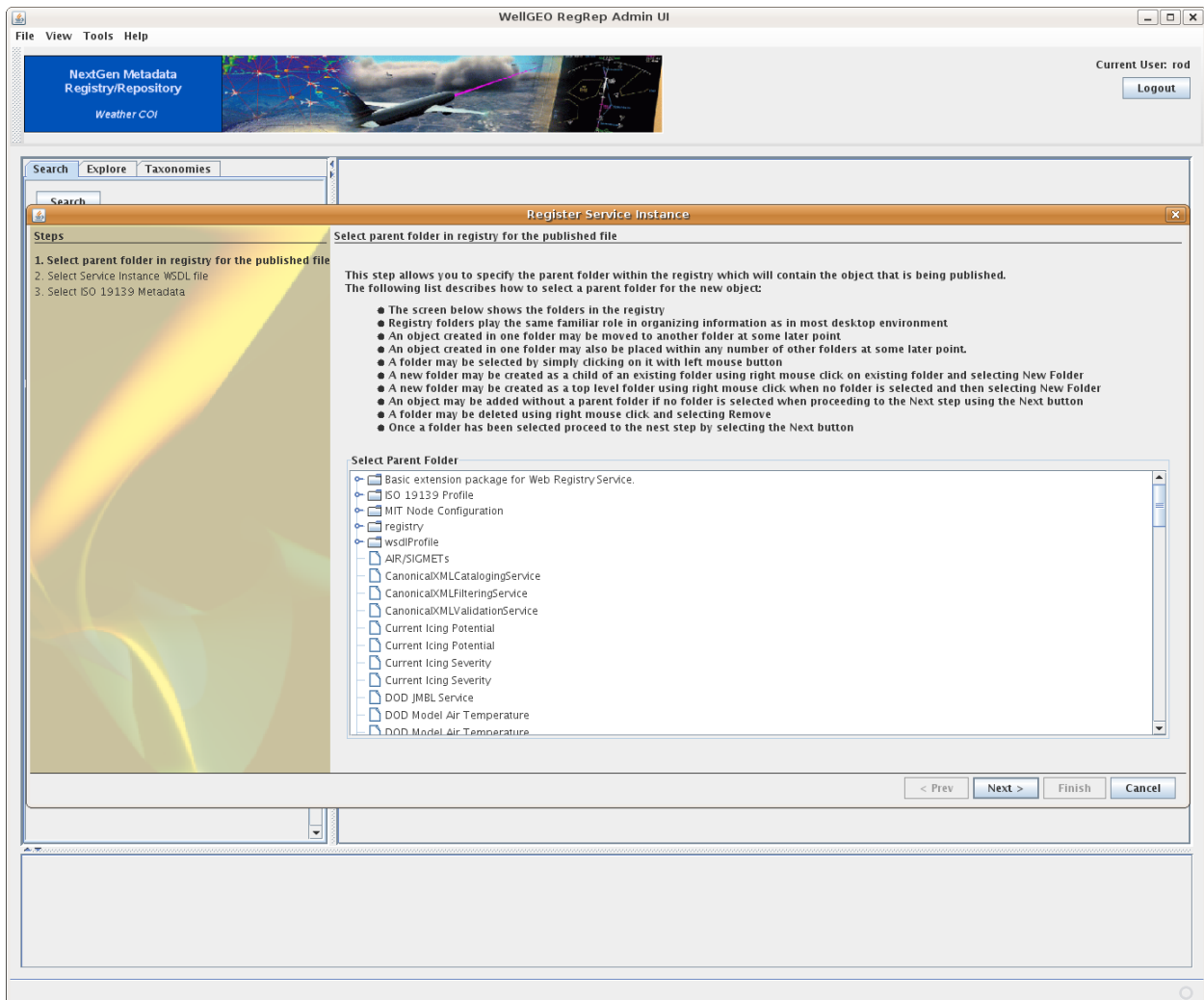


Figure 3.31: Service Publish - Select Parent Folder

4. Click on the Next button to skip the optional step 1 (“**Select Parent Folder**”) and go to step 2 (“**Select Service Instance WSDL file**”).

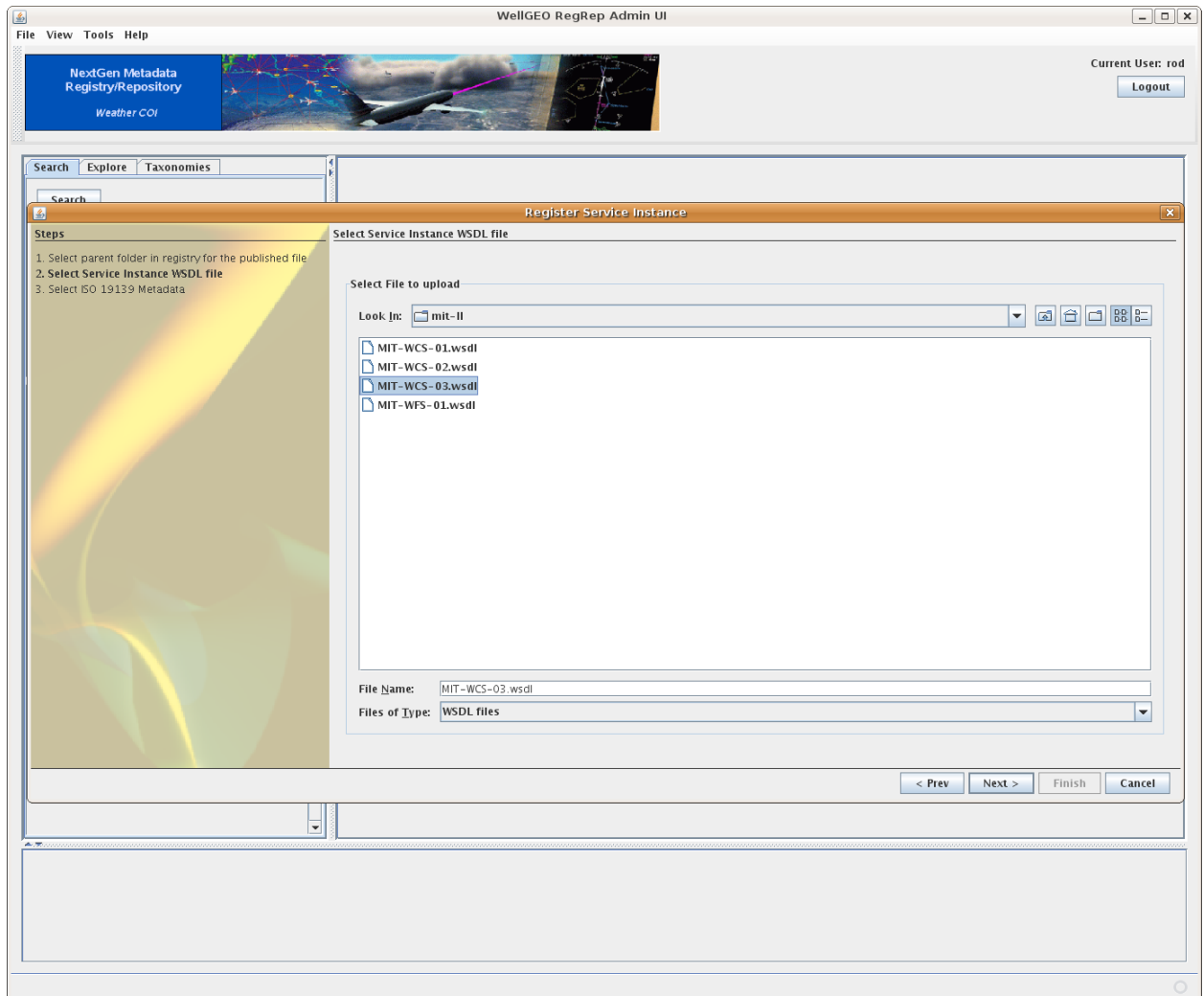


Figure 3.32: Publish Service Instance - Select WSDL File

5. Specify the WSDL file for the service instance. Browse to \$TEST_DATA_DIR/wxcube-metadata/services/mit-ll, and select the appropriate WSDL file: MIT-WCS-03.wsdl.
6. Select Next to go to “**Select ISO 19139 Metadata**” step

- Specify the ISO metadata file. Browse to \$TEST_DATA_DIR/wxcube-metadata/services/mit-ll, (this should still be there by default from the previous step) and select the appropriate ISO 19139 file: iso-metadata-service-MIT-WCS-03.xml

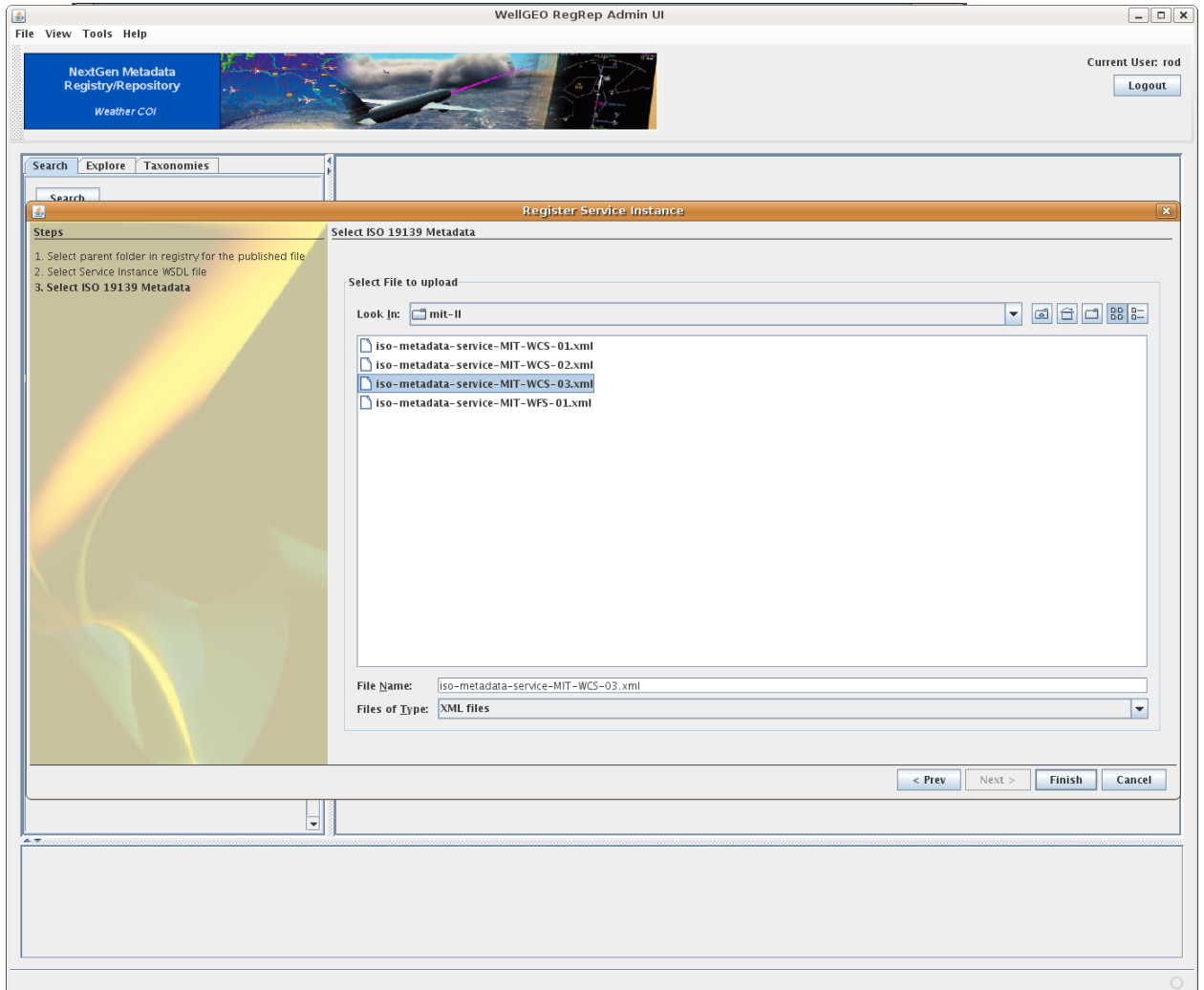
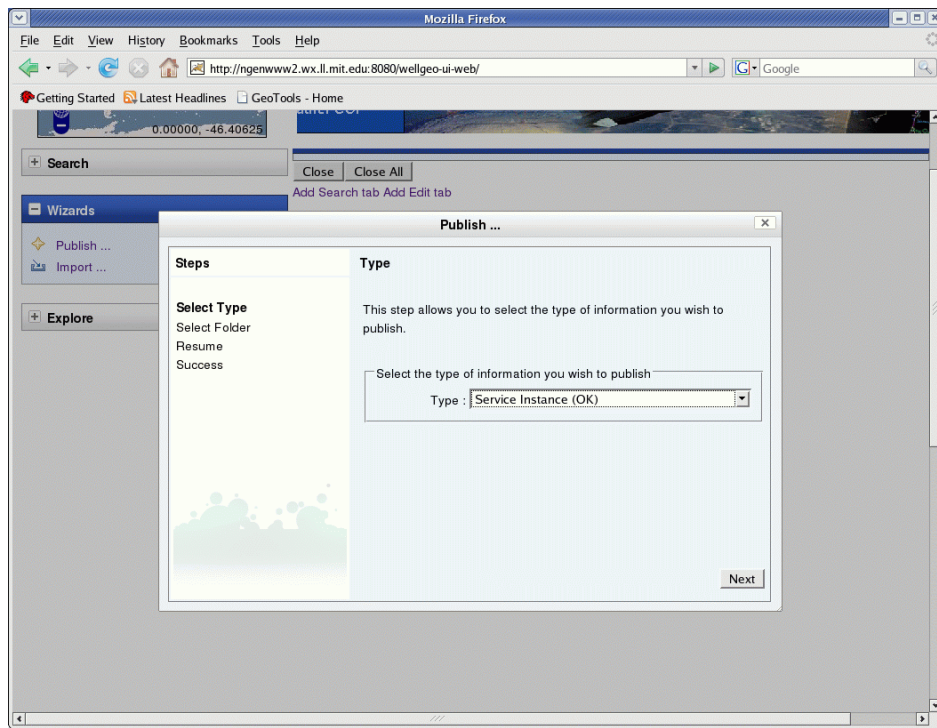
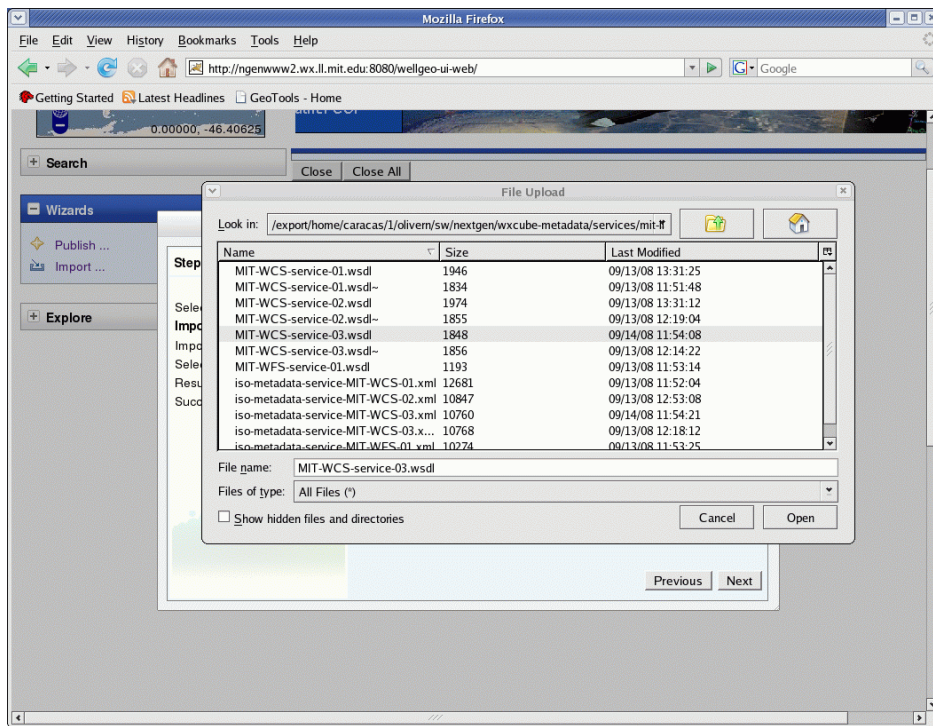
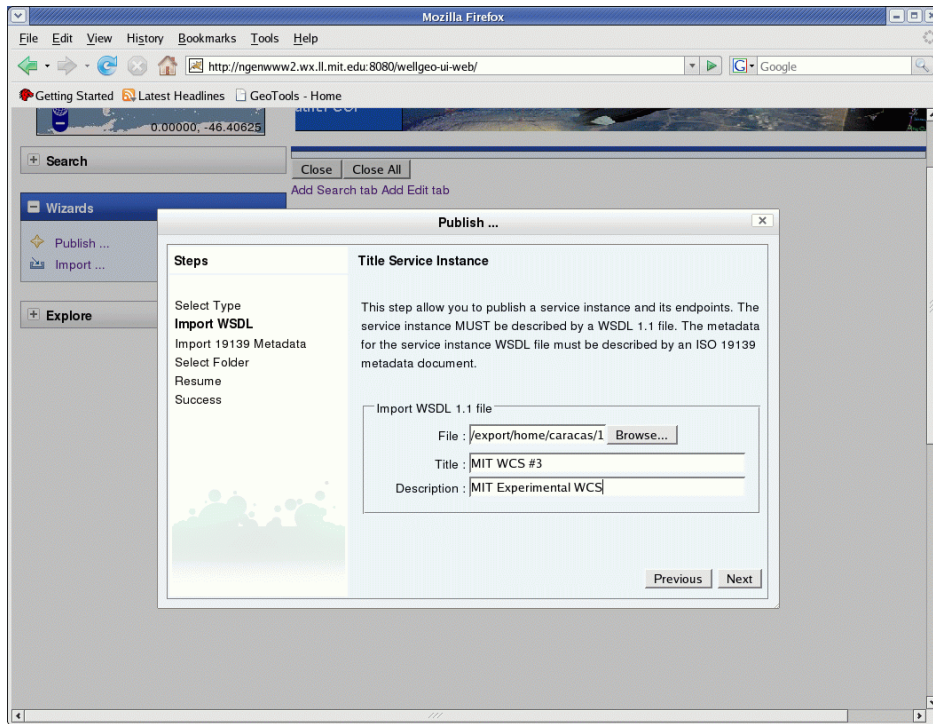
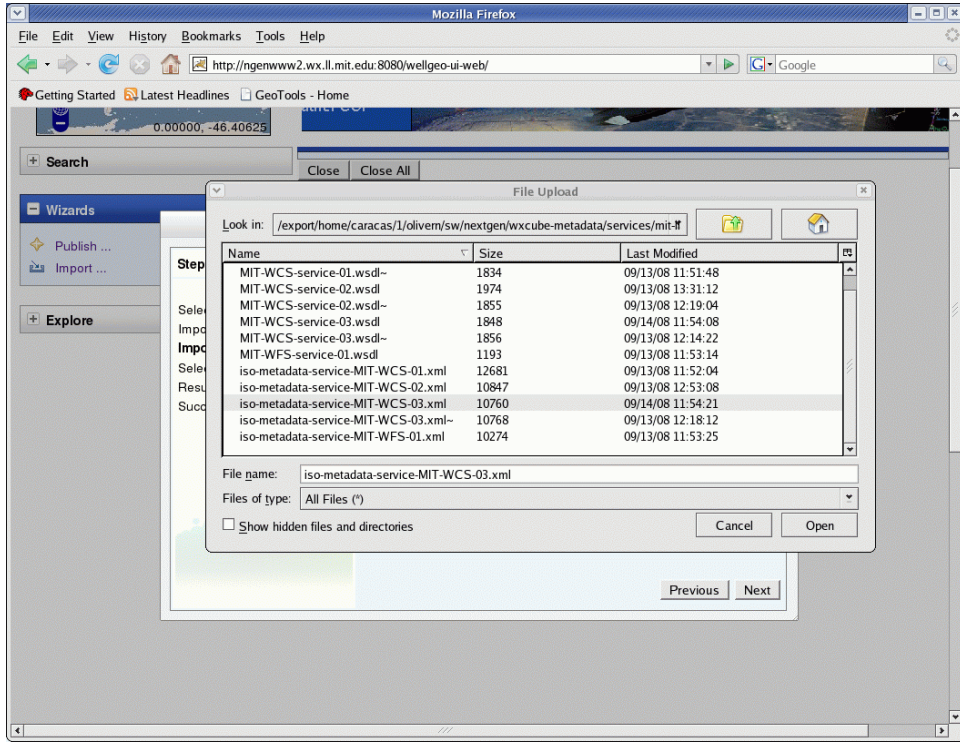


Figure 3.33: Publish Service Instance - Select ISO 19139 Metadata File









8. Click on the “**Finish**” button to publish the service instance specified by the WSDL and the ISO 19139 metadata file. An indication of success is returned in the Summary page.
9. Verify that the service was successfully published by searching for all services in the registry. Added to the former services should be one with the name MIT_WebCoverageService-03, with a description that includes an indication that it is the new experimental service.

3.7 Fault Tolerance Support in Registry Client API

This section verifies the fault tolerance capabilities of the registry client API available from the WxForge regrep4-client project support fault tolerant registry access. This test assumes that the regrep4-client project’s trunk tree has been checked out of WxForge svn repository with its root directory at \$REGREP4_CLIENT.

The tests in this section use a command line Java program called RegistryTestClient which performs a set of queries against its target registry. The RegistryTestClient program uses the regrep4-client Java client library developed by MIT Lincoln Labs. A specific benefit of using the regrep4-client library is that it provides the ability for a registry client to specify an infinite chain of backup registries in addition to a target registry. The regrep4-client Java client library first attempts to send the request to the target registry which is first in the back chain. If the target registry is unavailable or if the connections times out after a configurable time period then the library must re-route the request to the next registry in the backup chain. This process is repeated until a registry in the backup chain is able to respond to the request in the configured timeout period.

A client program specifies the backup registries and timeout period using a configuration file. For the RegistryTestClient this configuration file is specified via the “-c” option. For the following test we use a configuration file that specifies:

- A backup chain consisting of the following sequence of registries:
 1. MIT/LL-Primary – A registry representing the primary MIT-LL registry. It is configured with an [incorrect URL with a nonexistent host](#) to simulate the scenario that the registry is unavailable.
 2. MIT/LL-Backup1 - A registry representing the secondary or backup MIT-LL registry. It is configured with an [incorrect URL with a non-existent port](#) to simulate the scenario that the registry is unavailable.
 3. FAA/Backup1 – A registry representing the FAA registry that contains a union of data in MIT-LL and NWS registries and serves as backup for both of them
 - 20 seconds as timeout period for all registries in the back up chain
1. In a shell window change directory to: \$REGREP4_CLIENT/client

2. Type the following command to run the RegistryTestClient test with the specified configuration file:

```
./RegistryTestClient -c src/test/resources/registry-config-faultTolerant.xml | tee /tmp/test.log
```

Note that the configuration file specifies that client requests are to be sent to the MIT-LL registry and if it is not available are to be rerouted to a backup registry which is the FAA registry. The URL for the MIT-LL registry is deliberately incorrect to simulate the MIT-LL registry to be unavailable.

The output of the command is logged in file /tmp/test.log

3. Read file /tmp/test.log and verify that the following registries are blacklisted as they are unreachable:

Blacklisting registry 'MIT/LL-Primary'

... text omitted for brevity ...

Blacklisting registry 'MIT/LL-Backup1'

4. Jump to the end of the /tmp/test.log and verify that no errors are reported and the final output looks like:

/DataCube/Unrestricted/Regulatory/Commercial

Registry not writeable - done

4. Implementation Verification – Web Coverage Service Reference Implementation (WCSRI)

The Web Coverage Service Reference Implementation (WCSRI) is used to provide gridded data (i.e., coverages) access and services for the NNEW 4-D Weather Data Cube. This capability was included in previous fiscal year NNEW demonstrations, however, the WCSRI has been completely rewritten for FY '09. The WCSRI work this year has focused on requirements analysis, design and implementation of this primary NNEW 4D Weather Data Cube component.

The WCSRI is intended to provide an “out-of-the-box” solution for gridded data access services. The WCSRI is completely written in Java and therefore has no explicit platform or hardware requirements. It provides a simple configuration mechanism to allow Service Providers to specify the coverages they wish to serve, coverage data directory paths, coverage identifiers, coverage metadata and other coverage specific information. In addition, the WCSRI provides coverage subsetting capabilities by field, time and geographical constraints.

The test cases in this chapter will demonstrate the following capabilities:

- WCSRI Administration
 1. Adding and configuring a coverage
 2. FUSE Servicemix Integration
- WCSRI Core Services
 1. GetCapabilities, DescribeCoverage and GetCoverage Operations
 2. Netcdf 4 Responses

A number of different client applications will be used to perform the verification, as follows:

- Maven and SoapUI plugin (black box test suite)
- SoapUI GUI
- ToolsUI
- NNEW FY09 Integrated Java Application

4.1 Test Environment and Setup

4.1.1 Dependency Installations

The tests contained in this chapter have a number of dependencies on various packages that must be installed prior to running these tests successfully. These specific prerequisite dependencies are listed below and are described in Section 2, Test Requirements. Additional details are included below where necessary.

- **Memory**

The test computer should have a minimum of 1048M of RAM

- **Internet Connection**

- **Java Web Start**

- **Apache Maven**

Version 2.1.0 minimum. Verify that your Maven settings.xml file contains the following:

```
<profile>
  <id>mitProfile</id>
  <repositories>
    <repository>
      <id>mitRepo</id>
      <name>MIT Wxforge Repository </name>
      <url>http://wxforge.wx.ll.mit.edu:8080/archiva/repository/local</url>
    </repository>
  </repositories>
</profile>

<activeProfiles>
  <activeProfile>mitProfile</activeProfile>
</activeProfiles>
```

- **Fuse ESB 4.1.0.2 (ServiceMix)**

Verify that the \$FUSE_ESB_HOME/etc/org.apache.servicemix.features.cfg file contains the following entries in the featuresBoot property:

```
featuresBoot=activemq,camel,jbi-cluster,web,servicemix-cxf-bc,servicemix-file,servicemix-ftp,servicemix-http,servicemix-jms,servicemix-mail,servicemix-bean,servicemix-camel,servicemix-cxf-se,servicemix-drools,servicemix-eip,servicemix-osworkflow,servicemix-quartz,servicemix-scripting,servicemix-validation,servicemix-saxon,servicemix-wsn2005
```

- **HDF5**

Verify that the hdf5 libraries are accessible in any of the directories listed in the \$LD_LIBRARY_PATH variable.

- **Netcdf 4 C Libraries**

Verify that the netcdf 4 libraries are accessible in any of the directories listed in the \$LD_LIBRARY_PATH variable.

- **MIT Lincoln Labs Netcdf JNI Libraries**

Verify that the liblnetcdf.so library file is available in any of the directories listed in the \$LD_LIBRARY_PATH variable.

- **SoapUI**

At a minimum, SoapUI version 3.0.x must be installed on the system, in a directory henceforth referred to as \$SOAPUI_HOME. To run SoapUI, first verify that the \$SOAPUI_HOME/bin/soapui.sh is executable, and then type the following command (replacing the \$SOAPUI_HOME variable as appropriate):

```
$SOAPUI_HOME/bin/soapui.sh
```

There is a known issue with SoapUI, in that it may immediately crash on startup when run in a Linux environment. To date, the fix for this issue is to edit the soapui.sh file in \$SOAPUI_HOME/bin. Verify or edit the line containing the *first reference* (i.e., the declaration of) to the “JAVA_OPTS” property, such that it contains the “-Dsoapui.jxbrowser.disable=true” entry as below. In addition, verify that the -Xmx memory option is at least 384m:

```
JAVA_OPTS="-Xms128m -Xmx384m -Dsoapui.properties=soapui.properties -  
Dsoapui.home=$SOAPUI_HOME -Dsoapui.jxbrowser.disable=true"
```

SoapUI also requires a number of jar files to be placed in the \$SOAPUI_HOME/bin/ext directory in order to successfully execute the tests within the WCSRI SoapUI project. From the steps as described in the WCSRI Installation section below, copy the following bundle jar files into the \$SOAPUI_HOME/bin/ext directory:

```
$WCSRI_HOME/bundles/gds-pojo-service-[version].jar
```

```
$WCSRI_HOME/bundles/wcs-service-[version].jar
```

In addition, copy the gds-pojo-service-[version].jar to a temporary directory, and extract the file “netcdf-api-4.0.48.jar” from the gds-pojo-service-[version].jar by typing:

```
jar xvf gds-pojo-service-[version].jar
```

Copy the extracted “netcdf-api-4.0.48.jar” into the \$SOAPUI_HOME/bin/ext directory.

- **Ncdump**

The netcdf 4 version of ncdump is required.

- **ToolsUI**

ToolsUI is a Java Web Start application that will be used to validate the Netcdf 4 data returned from the WCSRI.

- **WCSRI Installation**

The tests in this chapter verify the correct configuration and functioning of the WCSRI, but do not include the installation as part of the test plan. Rather, the installation of the WCSRI is a prerequisite.

Installing the WCSRI requires you to download the following two files from <https://wiki.ucar.edu/display/NNEW/Releases>:

- wcsri-[version]-release.zip [or tar.gz]
- wcsri-[version]-validation.zip [or tar.gz]

Extract the files into a directory, referred to as \$WCSRI_HOME in this document. Follow the directions as specified in \$WCSRI_HOME/INSTALLATION.txt. You may skip the steps involving starting ServiceMix as well as the validation steps, as these will be performed as part of the test plan. However, verify the following prior to moving on to the tests:

- Verify that the \$WCSRI_HOME/conf/wcsri.cfg has been copied to \$FUSE_ESB_HOME/etc.
- Verify that the \$FUSE_ESB_HOME/etc/wcsri.cfg file has the [wcs.rootDataDir] property set to the root data directory for your WCSRI instance. Verify that the root data directory exists and has appropriate read and write privileges. In the remainder of this document, this directory will be referred to as \$WCSRI_ROOT_DATA_DIR.
- Verify that the \$WCSRI_HOME/conf/servicesMeta.xml file has been copied to \$WCSRI_ROOT_DATA_DIR.
- Verify that the \$FUSE_ESB_HOME/etc/wcsri.cfg file has the [wcs.endpoint] property set to the exposed endpoint for your WCSRI instance. This endpoint will be referred to as \$WCSRI_ENDPOINT in the remainder of this document.

4.2 WCSRI Administration

There are several steps one should take to verify the installation of the WCSRI. Though some of the following information is duplicated from Section 4.1.1, these test steps illustrate the configurability of the server.

First, edit the \$FUSE_ESB_HOME/etc/wcsri.cfg file, and verify the following configurable property settings:

- [wcs.rootDataDir] property is set to the root data directory for your WCSRI instance. This will be referred to as \$WCSRI_ROOT_DATA_DIR.
- [wcs.tempDir] property is set to a valid temporary directory. The WCSRI will use this to write temporary data files.
- [wcs.servicesMetaFile] property is set to the name of a valid services metadata file. Also, verify that the named file exists in the \$WCSRI_ROOT_DATA_DIR.

- [wcs.endpoint] property is set to the exposed endpoint for your WCSRI instance. This will be referred to as \$WCSRI_ENDPOINT.
- [wcs.mtomEnabled] property is set to true. Further test steps require MTOM to be enabled.

4.2.1 Services Metadata

The WCSRI's GetCapabilities operation provides a high level description of the capabilities of the server. The description consists of metadata about the server, including detailed information regarding the Service Provider, a description of the exposed endpoints and a summary of the coverages served by the particular WCSRI instance. In this section, we will configure some sample metadata for an instance of the WCSRI.

Edit the [wcs.servicesMetaFile] found in the \$WCSRI_ROOT_DATA_DIR. If you originally copied the file from the \$WCSRI_HOME/conf distribution directory, then it should initially look similar to **Error! Reference source not found..** Modify the content of the “ServiceIdentification” and “ServiceProvider” elements to specify the details of your organization. For example, set the “ProviderName,” “IndividualName,” and “ContactInfo” appropriately. In addition, (optionally) search and replace the “http://mymachine:9090/wcs” entries with your \$WCSRI_ENDPOINT.

```

servicesMeta x
1 <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2 <Capabilities xmlns="http://www.opengis.net/wcs/1.1"
3   xmlns:ows="http://www.opengis.net/ows/1.1"
4   xmlns:gml="http://www.opengis.net/gml"
5   xmlns:xlink="http://www.w3.org/1999/xlink" version="1.1.1">
6   <ows:ServiceIdentification>
7     <ows:Title>NNEW Web Coverage Service Version 1.0 (WCS)</ows:Title>
8     <ows:Abstract>This is a trimmed down implementation of the WCS 1.1.2 specification (see
9       www.opengis.net/wcs/1.1.2). The intent of the server is to provide gridded
10      data access services for aviation weather, as part of the NextGen Network
11      Enabled Weather (NNEW) program.</ows:Abstract>
12     <ows:ServiceType>WCS</ows:ServiceType>
13     <ows:ServiceTypeVersion>1.1</ows:ServiceTypeVersion>
14     <ows:Fees>None</ows:Fees>
15     <ows:AccessConstraints>none</ows:AccessConstraints>
16   </ows:ServiceIdentification>
17   <ows:ServiceProvider>
18     <ows:ProviderName>NextGen Network Enabled Weather (NNEW)</ows:ProviderName>
19     <ows:ServiceContact>
20       <ows:IndividualName>Rob Weingruber</ows:IndividualName>
21       <ows:PositionName>Software Engineer</ows:PositionName>
22       <ows:ContactInfo>
23         <ows:Phone>
24           <ows:Voice>303-497-2850</ows:Voice>
25         </ows:Phone>
26         <ows:Address>
27           <ows:DeliveryPoint>FL2</ows:DeliveryPoint>
28           <ows:DeliveryPoint>3300 Mitchell Lane</ows:DeliveryPoint>
29           <ows:City>Boulder</ows:City>
30           <ows:AdministrativeArea>Colorado</ows:AdministrativeArea>
31           <ows:PostalCode>80305</ows:PostalCode>
32           <ows:Country>U.S.</ows:Country>
33           <ows:ElectronicMailAddress>weingrub[at]ucar[dot]edu</ows:ElectronicMailAddress>
34         </ows:Address>
35         <ows:OnlineResource xlink:href="http://ral.ucar.edu/projects/nnew/">
36           <ows:HoursOfService>8x5x365</ows:HoursOfService>
37         <ows:ContactInstructions>email</ows:ContactInstructions>
38       </ows:ContactInfo>
39       <ows:Role>Lead Engineer</ows:Role>
40     </ows:ServiceContact>
41   </ows:ServiceProvider>
42   <ows:OperationsMetadata>
43     <ows:Operation name="GetCapabilities">
44       <ows:DCP>
45         <ows:HTTP>
46           <ows:Post xlink:href="http://mymachine:9090/wcs">
47             <ows:Constraint name="SOAP"/>
48           </ows:Post>
49         </ows:HTTP>

```

Figure 4.1: Sample servicesMeta.xml File

4.2.2 Configuring a New Coverage

In order to validate the WCSRI, we will download, install and configure a sample data set. These instructions are in `$WCSRI_HOME/INSTALLATION.txt`, and are summarized here.

Extract the `wcsri-[version]-validation.zip` [or `tar.gz`] into a temporary directory. From the extracted content, copy the `wcsriValidation/testData/ruc-mdvConvert-nc4` subdirectory into your configured `$WCSRI_ROOT_DATA_DIR`. This new directory, named “`$WCSRI_ROOT_DATA_DIR/ruc-mdvConvert-nc4`” and referred to as `$WCSRI_TEST_COVERAGE_DIR`, contains XML configuration files for a test dataset (i.e., coverage) as well as a number of weather data files within several subdirectories. The content of the `$WCSRI_TEST_COVERAGE_DIR` should look similar to Figure 4.2: Listing of `WCSRI_TEST_COVERAGE_DIR`.

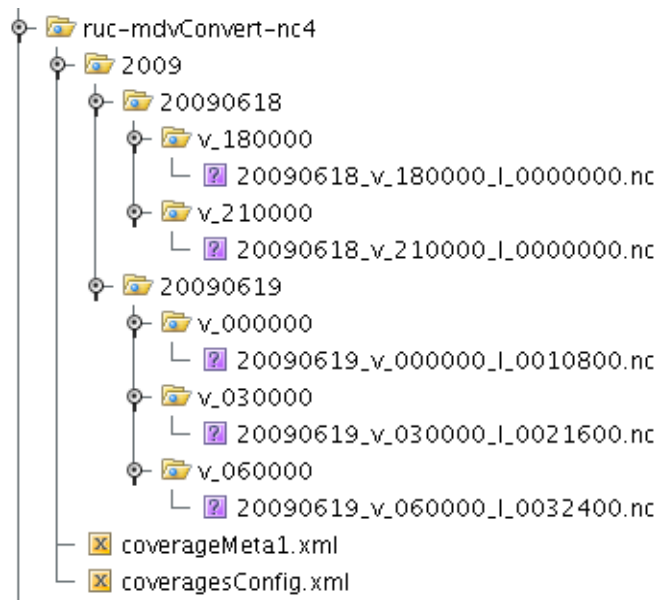


Figure 4.2: Listing of `WCSRI_TEST_COVERAGE_DIR`

Look at the content of the `coveragesConfig.xml` particular coverage (i.e., contains configuration information regarding the below the `$WCSRI_TEST_COVERAGE_DIR` convention. Since the black box tests executed in coverage, do not edit this file. However, notice that Review the following entries (for informational

file, which is located in the root directory for this `$WCSRI_TEST_COVERAGE_DIR`). This file test coverage dataset, whose weather data files exist with a specific directory and file naming subsequent test steps depend on this specific the content of that file is similar to Figure 4.3. purposes only):

- `fileExtension` – the file extension for the weather data files considered as part of this coverage
- `Id` - the unique identifier for this coverage
- `Field` - the fields of data available from this coverage

- CoverageMetadataFile – the name of the coverage metadata file, that must exist in this \$WCSRI_TEST_COVERAGE_DIR. Verify that the name is “coverageMeta1.xml”.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <DataSource type="FileSystem" fileExtension="nc">
4
5 <!-- ValidTimesType is optional. type="TimeList" is the default. -->
6 <!--ValidTimesType type="TimeList"/-->
7 <ValidTimesType type="TimeRange" resolutionSeconds="10800"/>
8
9 <Jani tor enabled="false">
10 <!-- Cron pattern format: Sec Min Hour DoM Mon DoW (Year) -->
11 <ScrubFrequency cronPattern="0 0 2 ? * *"/>
12 <!-- Duration format: "PnYnMnDnHnMnS"
13 "-P5Y2M10D" = five years, 2 months, 10 days ago
14 "-P15D" = fifteen days ago
15 "-PT15H" = 15 hours ago
16 "-PT2H30M" = 2 hours and 30 minutes ago
17 "-PT45M" = 45 minutes ago
18 -->
19 <KeepTime duration="-P15D"/>
20 </Jani tor>
21
22 <!-- You may list multiple Coverage elements if multiple coverages come from the same data files. -->
23 <Coverage>
24
25 <!-- The unique id of the coverage, taken from the NNEW Registry -->
26 <Id>urn:fdc:ncar.ucar.edu:Dataset:PoxDixonNetcdf4RUC20_Air_Temperature</Id>
27
28 <!-- The name of the WCS 1.1 compliant metadata file for this coverage, in this directory -->
29 <CoverageMetadataFile name="coverageMeta1.xml"/>
30
31 <!-- Multiple fields are allowed, but they must be homogenous wrt dimensions.
32 If they are not homogenous, then getCoverage requests will produce an error
33 for volume/corridor requests.
34 Field's name attribute is the name of the variable within the data files.
35 Field's id attribute will be used in the future. -->
36
37 <Field name="TMP" id="urn:fdc:ncar.ucar.edu:Field:Air_Temperature"/>
38
39 </Coverage>
40
41 </DataSource>
42

```

Figure 4.3: coveragesConfig.xml

Edit the coverageMeta1.xml file located in the \$WCSRI_TEST_COVERAGE_DIR. This file, shown in Figure 4.4: coverageMeta1.xml contains metadata regarding the test coverage dataset, and allows the Service Provider to enter metadata specific to their coverage. Much of this file is dynamically generated metadata (e.g., Identifier, Field, TemporalDomain), though other parts of the XML document may be edited or configured by the Service Provider. Change the contents of the “Title” and “Abstract” elements, and add an additional “Keyword” element.

As a result of this data set installation step, you have already configured a new coverage to be served by your WCSRI instance. Placing the weather data files (using the naming convention) in a subdirectory under the \$WCSRI_ROOT_DATA_DIR, and configuring the coveragesConfig.xml and coverageMeta1.xml files, will allow the WCSRI to automatically notice the offered coverage on startup.

```

coverageMeta1 x
1 <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2 <wcs:CoverageDescriptions xmlns:wcs="http://www.opengis.net/wcs/1.1"
3   xmlns:gml="http://www.opengis.net/gml"
4   xmlns:ows="http://www.opengis.net/ows/1.1"
5   xmlns:xlink="http://www.w3.org/1999/xlink">
6 <wcs:CoverageDescription>
7   <ows>Title>RUC Air Temperature (Pox Dixon Netcdf 4)</ows>Title>
8   <ows:Abstract>This dataset consists of RUC data for the Air Temperature 3D field. Its native
9
10  <ows:Keywords>
11    <ows:Keyword>RUC</ows:Keyword>
12    <ows:Keyword>Dixon</ows:Keyword>
13    <ows:Keyword>Pox</ows:Keyword>
14    <ows:Keyword>Netcdf4</ows:Keyword>
15  </ows:Keywords>
16
17  <ows:Metadata xlink:type="simple" about="http://weather.aero"/>
18
19  <!-- Do NOT edit! - Identifier will be filled out dynamically -->
20  <!--wcs:Identifier>urn:fdc:ncar.ucar.edu:Dataset:RUC20</wcs:Identifier-->
21
22  <wcs:Domain>
23    <wcs:SpatialDomain>
24      <ows:BoundingBox crs="urn:ogc:def:crs:EPSG:6.13:9801">
25        <ows:LowerCorner>-3332.14 -588.89</ows:LowerCorner>
26        <ows:UpperCorner>2783.58 3982.66</ows:UpperCorner>
27      </ows:BoundingBox>
28      <ows:BoundingBox crs="urn:ogc:def:crs:OGC:2:84">
29        <ows:LowerCorner>-126.227 16.168</ows:LowerCorner>
30        <ows:UpperCorner>-57.244 55.506</ows:UpperCorner>
31      </ows:BoundingBox>
32      <wcs:GridCRS>
33        <wcs:GridBaseCRS>urn:ogc:def:crs:EPSG:6.13:9801</wcs:GridBaseCRS>
34      </wcs:GridCRS>
35    </wcs:SpatialDomain>
36    <!-- Do NOT edit! - TemporalDomain will be added and filled out dynamically -->
37    <!--wcs:TemporalDomain-->
38    <!--</wcs:TemporalDomain-->
39  </wcs:Domain>
40
41  <wcs:Range>
42    <!-- Do NOT edit! - Field elements will be added and filled out dynamically -->
43    <!--wcs:Field-->
44    <!--</wcs:Field-->
45  </wcs:Range>
46
47  <!-- TODO SupportedCRS should be the CRS of the data, eg: WGS84+AltKm -->
48  <wcs:SupportedCRS>urn:ogc:def:crs:EPSG:6.13:9801</wcs:SupportedCRS>
49
50  <!-- Do NOT edit! - SupportedFormat will be filled out dynamically -->
51  <!--wcs:SupportedFormat>image/mdv</wcs:SupportedFormat-->
52  <!-- Additional meta data per WCS xsds -->
53
54 </wcs:CoverageDescription>
55 </wcs:CoverageDescriptions>
56

```

Figure 4.4: coverageMeta1.xml

4.2.3 Starting Fuse/ServiceMix

Open a console and change directory to \$FUSE_ESB_HOME. Verify that the \$LD_LIBRARY_PATH variable contains the location of the various needed libraries (e.g., libllnetcdf.so, the Netcdf4 libraries, and the HDF5 libraries). Verify that the gds-pojo-service-[version].jar and wcs-service-[version].jar has been

copied into the \$FUSE_ESB_HOME/deploy directory (from \$WCSRI_HOME/bundles). Start ServiceMix by typing:

```
./bin/servicemix
```

Verify that there are no exceptions thrown on startup. If there are, you may have to “**exit**” (or **Ctrl-C**) ServiceMix, and restart. ServiceMix uses new OSGI technology, and on occasion, has race-conditions on startup. This means that depending on the speed of loading certain bundles, ServiceMix may encounter an unresolved dependency. If that’s the case, simply restart ServiceMix and the problem usually disappears. In the ServiceMix console, type the following (in boldface):

```
smx@root: /> osgi
```

The ServiceMix console should now appear similar to Figure 4.5.

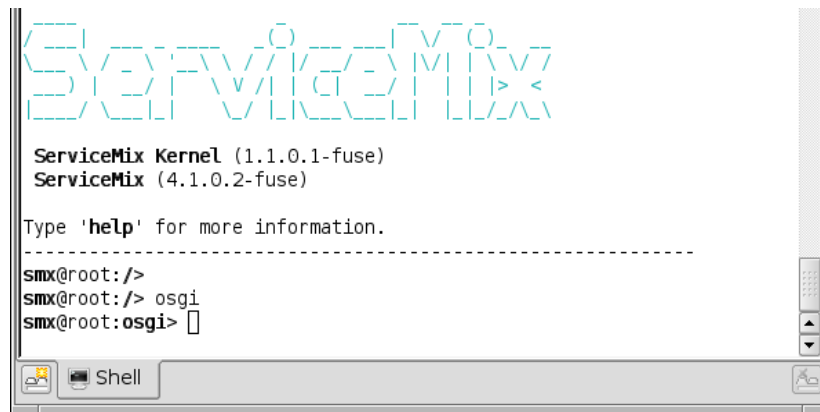


Figure 4.5: ServiceMix Console

In the ServiceMix console, type the following (in boldface):

```
smx@root:osgi> list | grep GDS
```

```
smx@root:osgi> list | grep WCS
```

Verify that the “**GDS POJO Service**” and “**WCSRI Web Service**” bundles have been installed and started successfully. The console should look similar to Figure 4.6, and “**Started**” should appear for each bundle. It takes several moments for ServiceMix to start the bundles, therefore, repeat the command until “**Started**” appears.

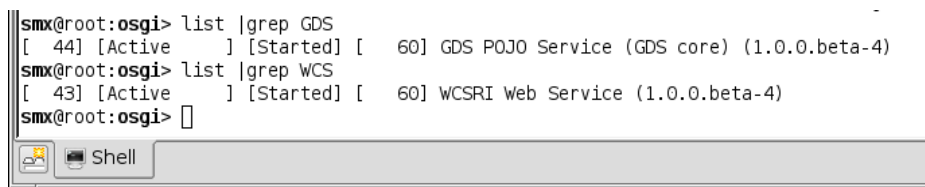


Figure 4.6: Successfully started bundles.

Switch to the “log” subshell by typing `log` from with the ServiceMix console:

```
smx@root:osgi /> log
```

Please note that depending on your log buffer settings for ServiceMix, you may or may not see the log test result described below. However, type the following command into the ServiceMix console:

```
smx@root:log /> d | grep DataSourceConfigurator
```

This should (depending on ServiceMix’s log settings) display the `$WCSRI_ROOT_DATA_DIR` that was found from the WCSRI configuration, as well as a summary of each coverage configuration found by the WCSRI. The logging information should look similar to Figure 4.7. Verify that the WCSRI is using the correct `$WCSRI_ROOT_DATA_DIR`, that the test data coverage was found with an identifier of “`urn:fdc:ncar.ucar.edu:Dataset:PoxDixonNetcdf4RUC20_Air_Temperature`”, and that “`TMP`” is an offered field.

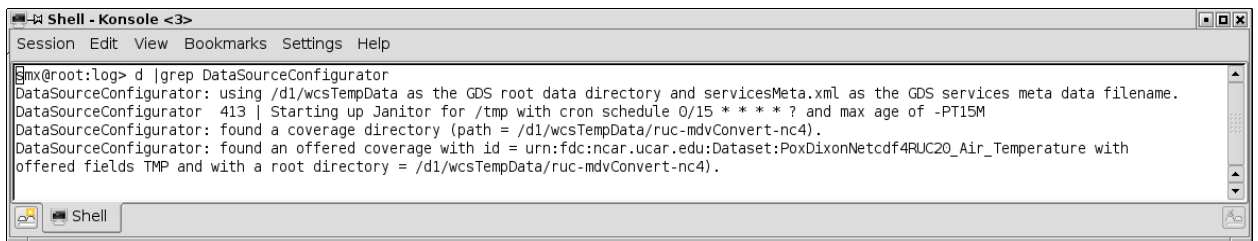


Figure 4.7: Log output

You should now have a successfully running WCSRI instance. The next set of tests will verify the exposed endpoint and the WCSRI functionality, and each *requires* that ServiceMix remains running.

4.3 WSDL Verification

If the WCSRI is running successfully, it’s a good idea to verify the validity of the exposed endpoint. Start a Firefox browser, and enter the URL of the `$WCSRI_ENDPOINT` followed by “`?wsdl`” into the location bar. Figure 4.8 shows an example URL in the location bar, as well as a sample of what the response should look like. Note that this test case demonstrates the WSDL capabilities of the WCSRI instance, and that different browsers may render the XML content differently than what you see in Figure 4.8.

```

- <definitions name="wcs" targetNamespace="http://www.opengis.net/wcs/wsd/1" >
- <types >
- <xsd:schema elementFormDefault="qualified" targetNamespace="http://www.opengis.net/wcs/1.1" version="1.1.2"
  xml:lang="en" >
  <xsd:include schemaLocation="http://granite:8888/nnew/fy09/wcs/soap?xsd=../schemas/wcs/1.1.2/wcsAll.xsd"/ >
  </xsd:schema >
</types >
- <message name="describeCoverageRequestMsg" >
  <part element="wcs:DescribeCoverage" name="parameters" > </part >
</message >
- <message name="getCapabilitiesResponseMsg" >
  <part element="wcs:Capabilities" name="parameters" > </part >
</message >
- <message name="getCoverageRequestMsg" >
  <part element="wcs:GetCoverage" name="parameters" > </part >
</message >
- <message name="getCapabilitiesRequestMsg" >
  <part element="wcs:GetCapabilities" name="parameters" > </part >
</message >
- <message name="exceptionReportMsg" >
  <part element="ows:ExceptionReport" name="parameters" > </part >
</message >
- <message name="getCoverageResponseMsg" >
  <part element="wcs:Coverages" name="parameters" > </part >
</message >
- <message name="describeCoverageResponseMsg" >
  <part element="wcs:CoverageDescriptions" name="parameters" > </part >
</message >
- <portType name="wcsPortType" >
- <operation name="getCapabilitiesOperation" >
  <documentation>Service definition of function getCapabilities</documentation >
  <input message="tns:getCapabilitiesRequestMsg" > </input >
  <output message="tns:getCapabilitiesResponseMsg" > </output >
  <fault message="tns:exceptionReportMsg" name="ExceptionReport" > </fault >
</operation >
- <operation name="describeCoverageOperation" >
  <documentation>Service definition of function describeCoverage</documentation >
  <input message="tns:describeCoverageRequestMsg" > </input >
  <output message="tns:describeCoverageResponseMsg" > </output >
  <fault message="tns:exceptionReportMsg" name="ExceptionReport" > </fault >
</operation >
- <operation name="getCoverageOperation" >
  <documentation>Service definition of function getCoverage</documentation >
  <input message="tns:getCoverageRequestMsg" > </input >
  <output message="tns:getCoverageResponseMsg" > </output >
  <fault message="tns:exceptionReportMsg" name="ExceptionReport" > </fault >
</operation >
</portType >
- <binding name="wcs" type="tns:wcsPortType" >
  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http/" >

```

Figure 4.8: WSDL Output

4.4 Verification using Maven and Black Box Testing

The WCSRI distribution contains a suite of black box tests that verify that the server is working correctly with the test dataset/coverage (as installed in Section 4.2.2). Though a description of the validation steps are outlined in the \$WCSRI_HOME/INSTALLATION.txt file, a summary is provided below since running the tests is a good indication that the server is working as intended. In addition, the black box tests will verify the GetCapabilities, DescribeCoverage and GetCoverage (for volumes and corridors) operations of the server, against the test data set as installed in Section 4.2.2.

Open a console (not ServiceMix!), and change directories to the \$WCS_VALIDATION_HOME/wcsriValidation/soapUI/wcsri-web-service/ directory. Carefully edit the pom.xml file in that directory. Search for the string "Service Providers", and change the "argument" element's URL to match the endpoint of your \$WCS_ENDPOINT, as configured in the wcsri.cfg file. For example:

```
<!-- Service Providers: Change the following URL to point to your $WCS_ENDPOINT -->
<argument>-e http://granite:8888/nnew/fy09/wcs/soap</argument>
```

Run the following Maven command from the console:

```
mvn org.codehaus.mojo:exec-maven-plugin:1.1.1:java >
myTestResults.txt
```

This will run a series of black box tests against the exposed endpoint of the WCSRI. The engine for the black box tests is a 3rd party application called SoapUI, and when run through Maven, the SoapUI-Plugin is being used to perform the tests. Each test case issues an XML request, over SOAP, to the WCSRI endpoint and then verifies that the response is correct. Some of the test cases expect valid SOAP responses, and others expect valid SOAP *fault* responses. By running the above command, the test output should be redirected into the file called “myTestResults.txt” in the same directory from which it was run.

Verify that the output indicates that each of the tests were successful. Do so by opening the file “myTestResults.txt” in a text editor, and search for status text similar to:

```
“Finished running soapUI testcase [GetVolume 1], time taken: 1653ms, status: FINISHED”
```

None of the tests should have status of “status: FAILED”.

4.5 Verification using SoapUI GUI and ToolsUI

In this test step we will run the SoapUI user-interface to look at and execute a few black box tests, instead of using the Maven plugin.

4.5.1 Starting SoapUI and Configuring the WCSRI Endpoint

The first step is to start SoapUI. Open a console and move to the following directory:

```
$WCSRI_VALIDATION_HOME/wcsriValidation/soapUI/wcsri-web-service
```

From that directory, start SoapUI by issuing the following command (replacing with the \$SOAPUI_HOME variable as appropriate):

```
$SOAPUI_HOME/bin/soapui.sh
```

On startup, SoapUI should look similar to Figure 4.9. Select the “**File**” → “**Import Project,**” and then import the project file, called WCSRI-112-soapui-project.xml, from the following directory:

```
$WCSRI_VALIDATION_HOME/wcsriValidation/soapUI/soapui/wcs112/robsSoapUI
```

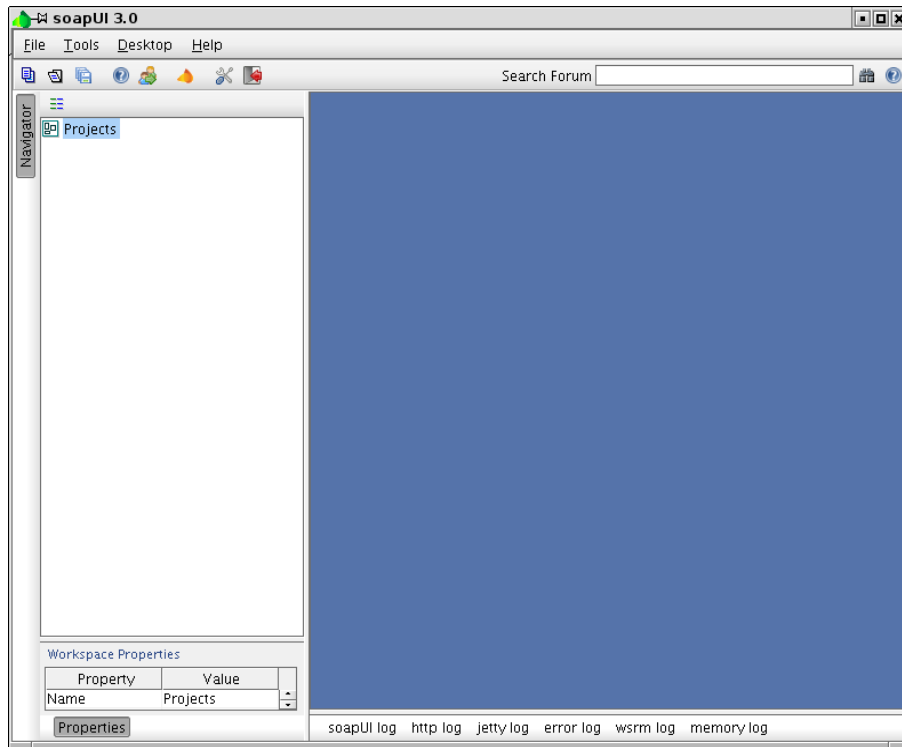



Figure 4.9: SoapUI GUI

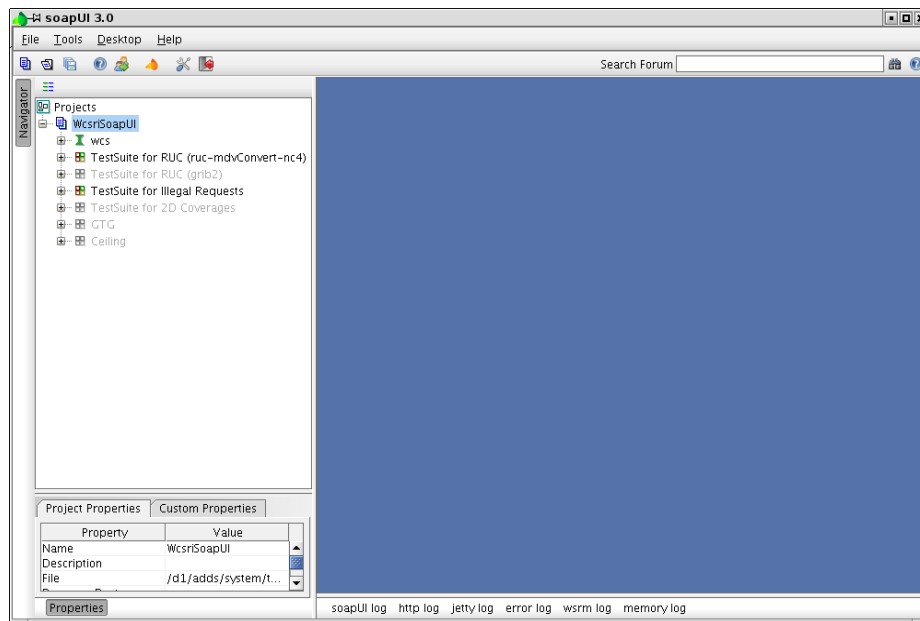




Figure 4.10: SoapUI Project

Once the project is imported, SoapUI should look similar to Figure 4.10. Double click on the green hourglass entry labeled “wcs” (i.e.,  **wcs**) and a new frame will appear. From within the new frame,

click on the tab labeled “**Service Endpoints.**” SoapUI should now look similar to Figure 4.11. Click on the “plus” button (i.e., ) to add a new endpoint for SoapUI, and set the URL for this new endpoint to be the value of your \$WCSRI_ENDPOINT. Select your new endpoint from the list (in the Service Endpoints tab), and click on the “**Assign**” button. Select “- **All Requests and TestRequests** -,” as in Figure 4.12. The test cases in the SoapUI project are now configured with the endpoint of your WCSRI instance.

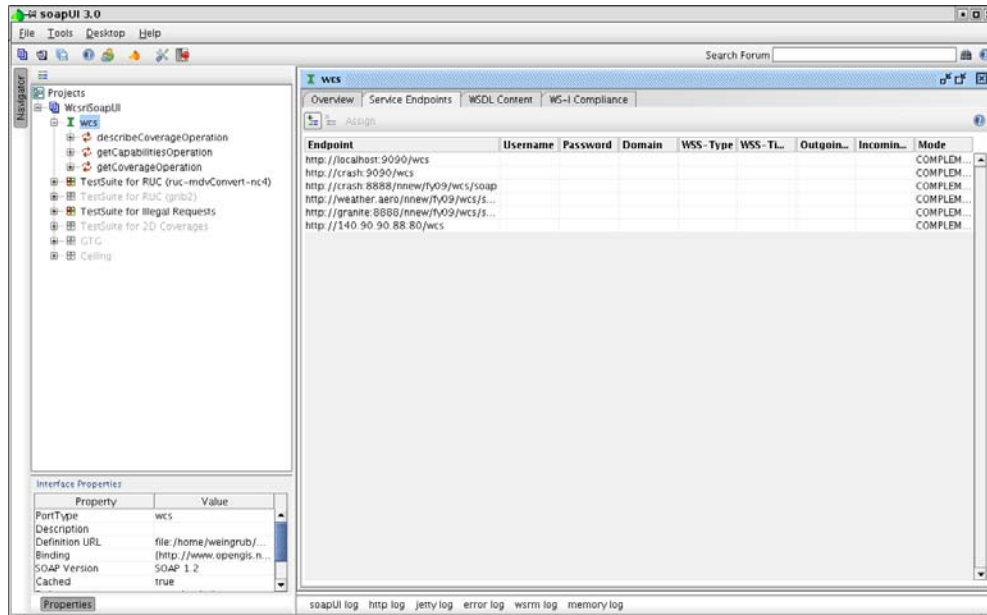


Figure 4.11: SoapUI Service Endpoints

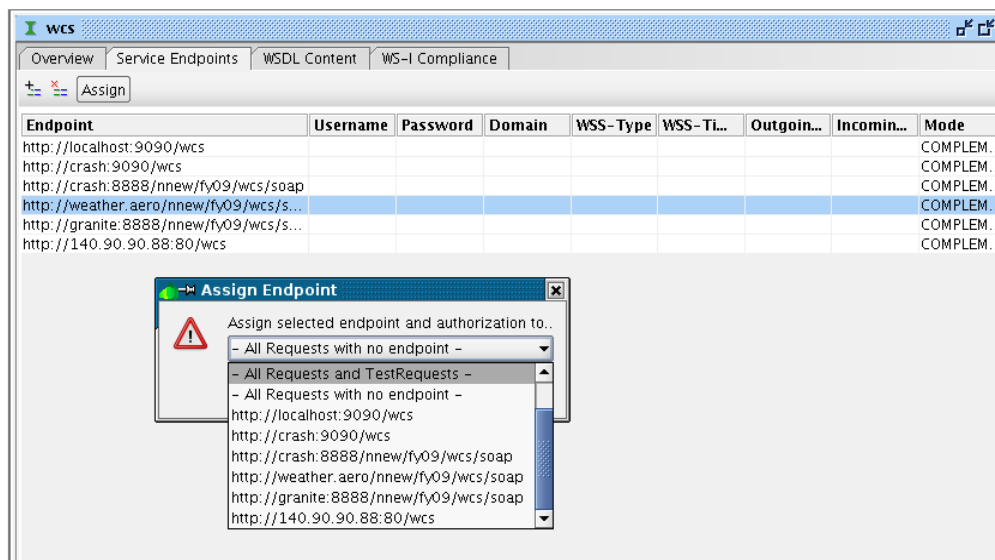



Figure 4.12: SoapUI - Assigning the endpoint.

4.5.2 Executing the Black Box Test Suite

Double click the “**TestSuite for RUC (ruc-mdvConvert-nc4)**” as listed on the left hand side of SoapUI. This will display a tree of test cases intended to validate the test dataset/coverage as configured in Section 4.2.2, and will open a new frame on the right hand side. Maximize the new frame. From the new frame, click on the green “play” button (i.e., ). All of the tests should be successful, and the test suite frame should now appear as in Figure 4.13.

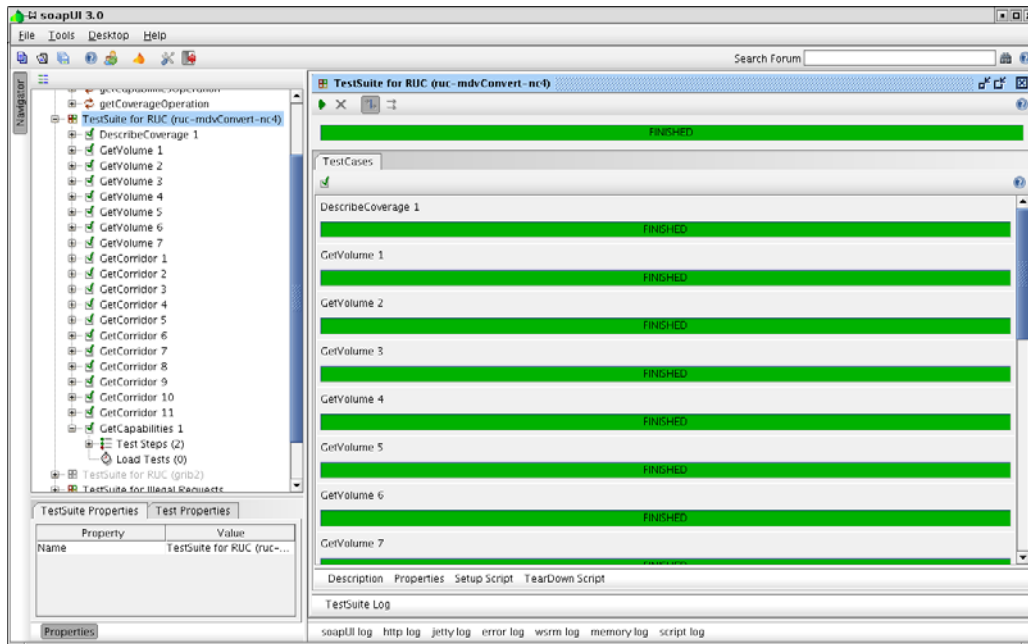




Figure 4.13: Successful Test Suite

4.5.3 Executing GetCapabilities

Double click on the “**GetCapabilities 1**” node, listed in the left-hand tree under the “**TestSuite for RUC (ruc-mdvConvert-nc4)**”, and maximize the new frame. Within the new frame, double click on the “**Test Request**” (i.e., ) and maximize the new frame. Within that new frame, execute the test by pressing the “play” button (i.e., ). The test should execute successfully, and SoapUI should now look similar to Figure 4.14. The “**Test Request**” frame should be divided in half (or have two tabs), where the left hand side contains the XML for a GetCapabilities request, and the right hand side contains the XML response (note that the SOAP headers have been hidden by SoapUI). Verify that the XML response contains the metadata modifications that were made to the servicesMeta.xml file, as configured in Section 4.2.1. This test illustrated a successful GetCapabilities request.

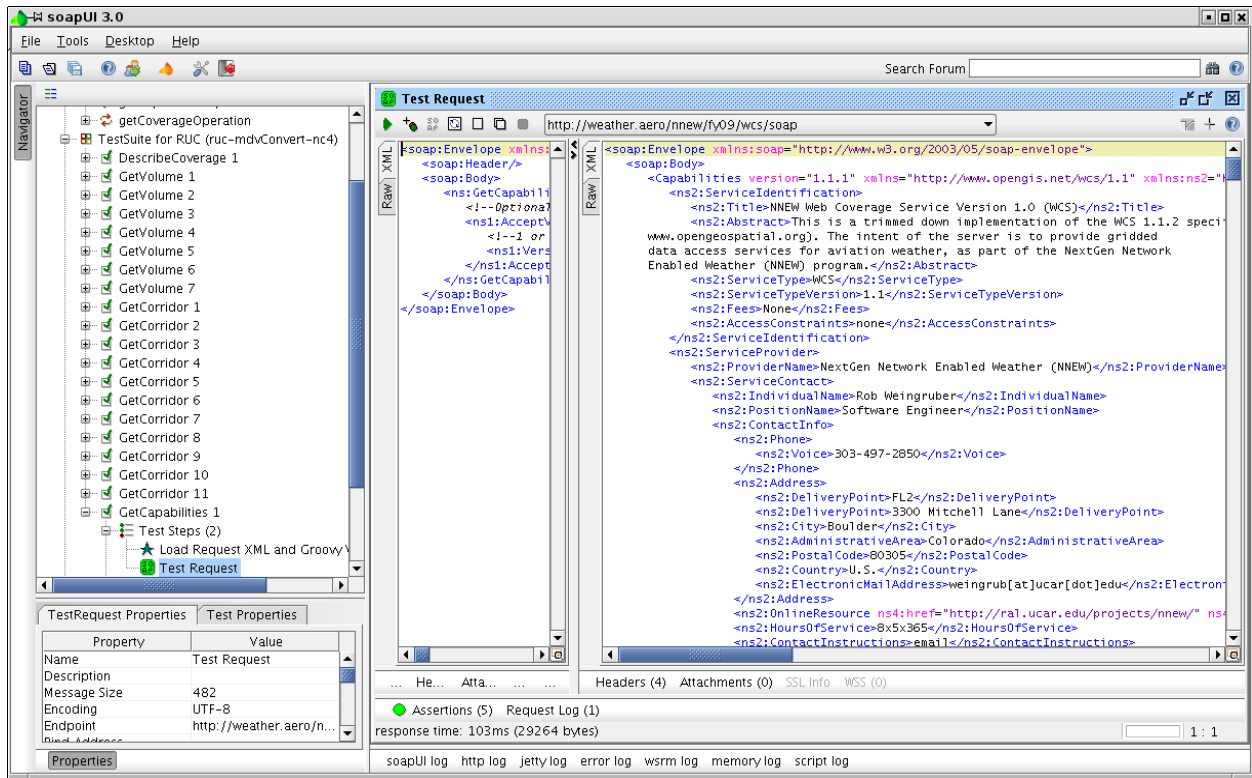




Figure 4.14: GetCapabilities 1 Request/Response

4.5.4 Executing DescribeCoverage

Double click on the “**DescribeCoverage 1**” node, listed in the left-hand tree under the “**TestSuite for RUC (ruc-mdvConvert-nc4)**”, and maximize the new frame. Within the new frame, double click on the “**Test Request**” (i.e., ) and maximize the new frame. Within that new frame, execute the test by pressing the “play” button (i.e., ) . The test should execute successfully, and SoapUI should now look similar to Figure 4.15. The “**Test Request**” frame should be divided in half (or have two tabs), where the left hand side contains the XML for a DescribeCoverage request, and the right hand side contains the XML response (note that the SOAP headers have been hidden by SoapUI). Note that the XML request contains the identifier of the test coverage/dataset, as configured in Section 4.2.2. Verify that the response contains the coverage metadata modifications that were made to the coverageMeta1.xml file, as configured in Section 4.2.2. Also note that “**TimePeriod**” contains the “**BeginPosition**” and “**EndPosition**” signifying the valid times (time period) of the coverage’s data availability. In addition, “**TMP**” is an available Field for the coverage. This test illustrated a successful DescribeCoverage request.

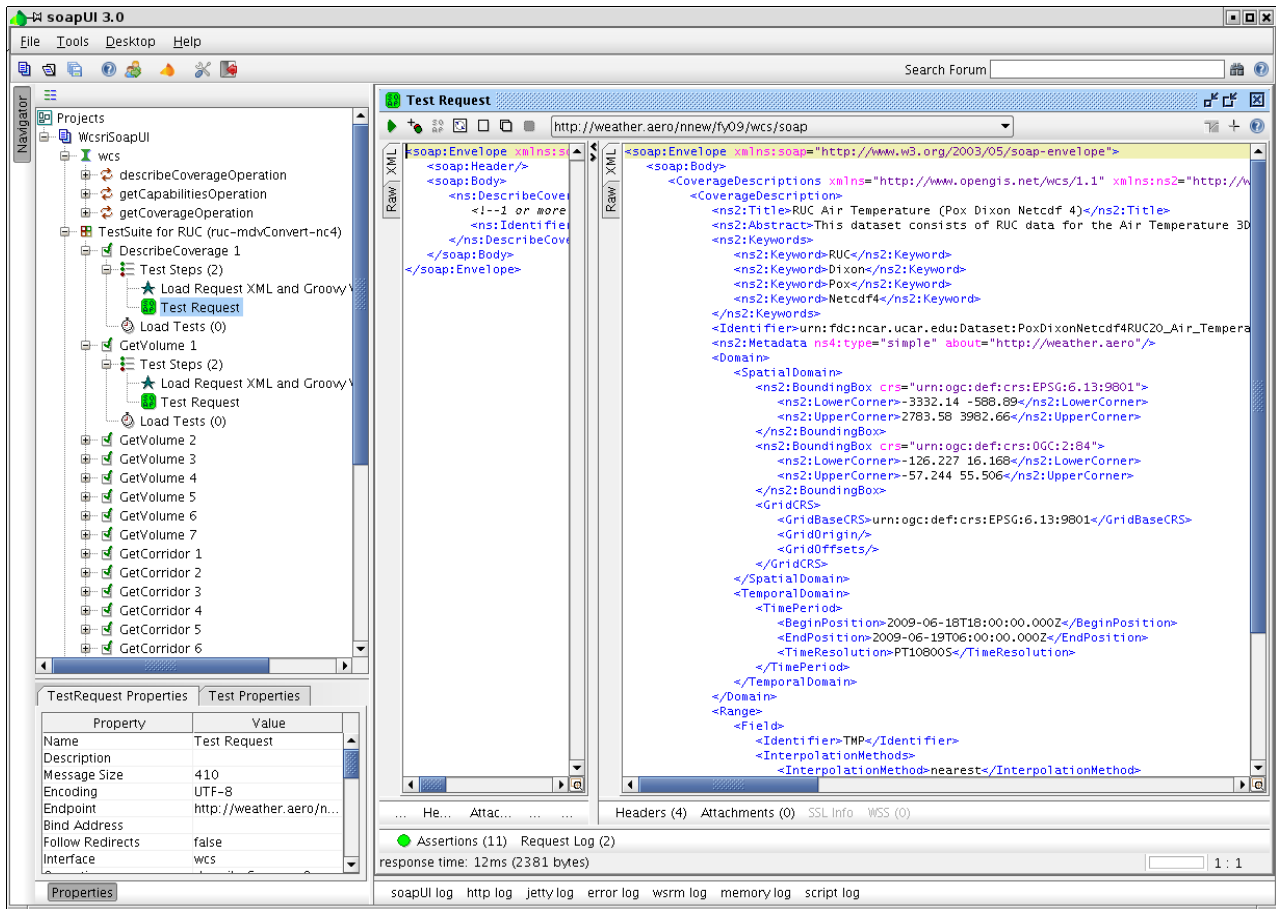




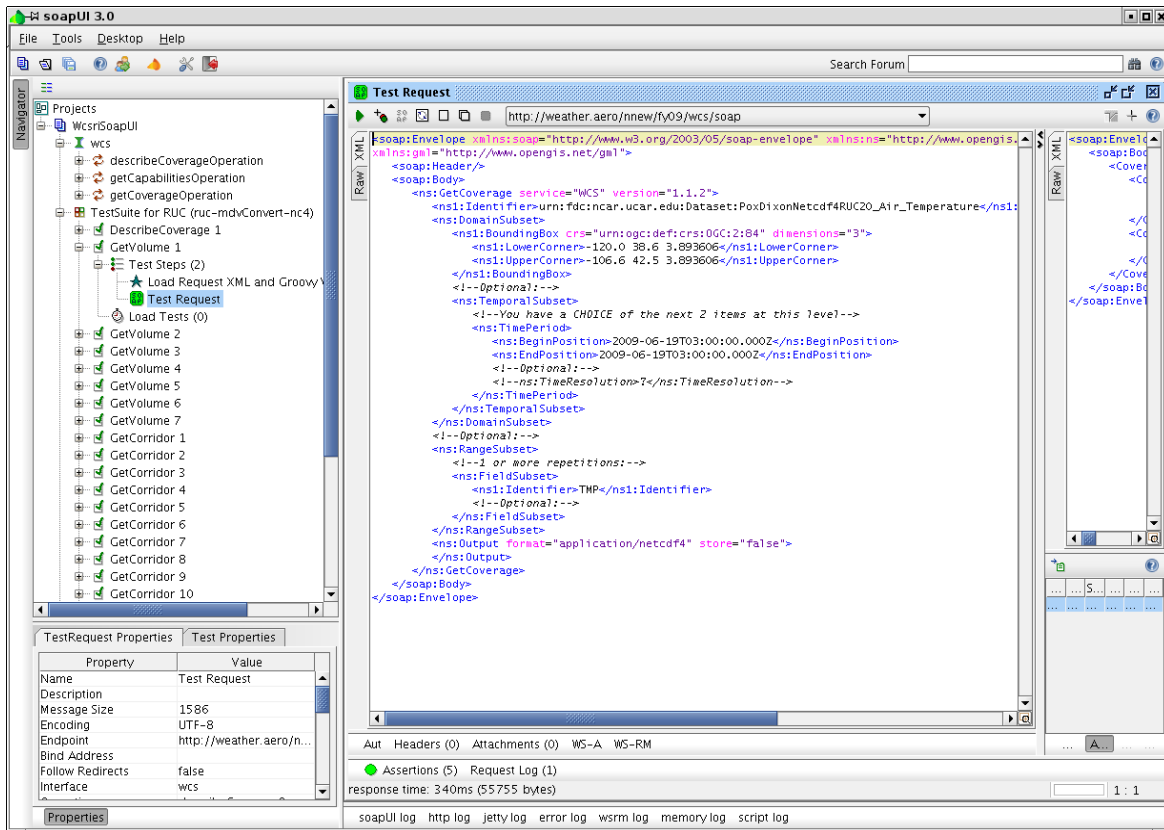
Figure 4.15: DescribeCoverage 1 Request/Response


4.5.5 Executing GetCoverage for a Volume

Double click on the “**GetVolume 1**” node, listed in the left-hand tree under the “**TestSuite for RUC (ruc-mdvConvert-nc4)**”, and maximize the new frame. Within the new frame, double click on the “**Test Request**” (i.e., ) and maximize the new frame. Within that new frame, execute the test by

pressing the “play” button (i.e., ). The test should execute successfully, and SoapUI should now look similar to the figure below. The “Test Request” frame should be divided in half (or have two tabs), where the left hand side contains the XML for a GetCoverage request, and the right hand side contains the XML response (note that the SOAP headers have been hidden by SoapUI). The figure below depicts the GetCoverage request for a volume of data. Note the following features of the request:

- The “**Identifier**” matches that of the test coverage dataset.
- The “**BoundingBox**” extends from a lower left of (-120.0, 38.6, 3.893606) to an upper right of (-106.6, 42.5, 3.893606) where the units are (longitude, latitude, altitude in km).
- The “**TemporalSubset**” for the request is “2009-06-19T03:00:00.000Z,” for which there is data available (according to the results of DescribeCoverage).
- The “**Identifier**” for the requested “**FieldSubset**” is “TMP”.



Verify that the response contains the Identifier as listed in the request, and that the response frame looks similar to Figure 4.16. Click on the “**Attachments**” tab near the bottom of the response frame, and you should see one attachment as shown in Figure 4.16. This illustrates the successful application of MTOM technology, which is similar to SOAP with Attachments. Select the single attachment, and just above it, click on the “**Save Attachment**” button (i.e., ). Save the file to “volume1.nc”. This test illustrated a successful GetCoverage request for a regular (single altitude) volume. [Keep SoapUI running].

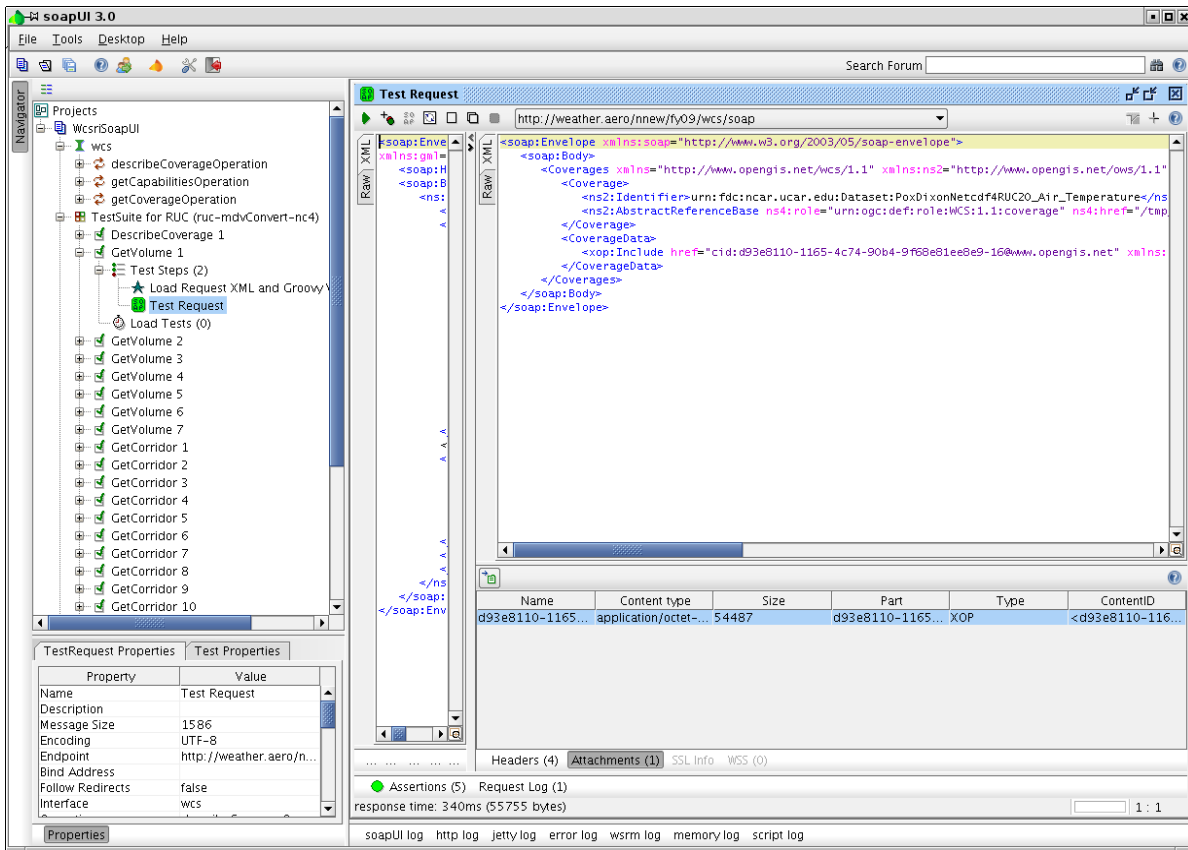





Figure 4.16: GetVolume 1 Response

4.5.6 Visualizing GetCoverage Volume Results

In this step, we will use a 3rd party Java Web Start application called ToolsUI to visualization the volume obtained in the last test step. The application is available from Unidata’s website, therefore, start Firefox and enter the following into the location bar:

<http://www.unidata.ucar.edu/software/netcdf-java>

The web page should appear as in Figure 4.17. Start ToolsUI version 4.0 by clicking on the first appearance of the words “launch from webstart” as shown in the bottom of Figure 4.17. After it downloads, ToolsUI should appear similar to Figure 4.18. Click on the “**FeatureTypes**” tab, then the “**Grids**” tab, and open the “volume1.nc” file (from the previous section) by clicking on the “Open Folder” button (i.e., ) to the right of the “dataset” text field. After the dataset is opened, select the “**Temperature**” or “**TMP**” grid as shown below. Click on the “Red Alien” button (i.e., ) to open another window, called the “**Grid Viewer**” similar to Figure 4.19 (though when it first appears, it will be black-ish). In the Grid Viewer, click on the “Red Alien” button (i.e., ) to visualize the volume. Zoom out as necessary. This test case illustrated visualization of Netcdf 4 volume data returned as a result of issuing a GetCoverage request.

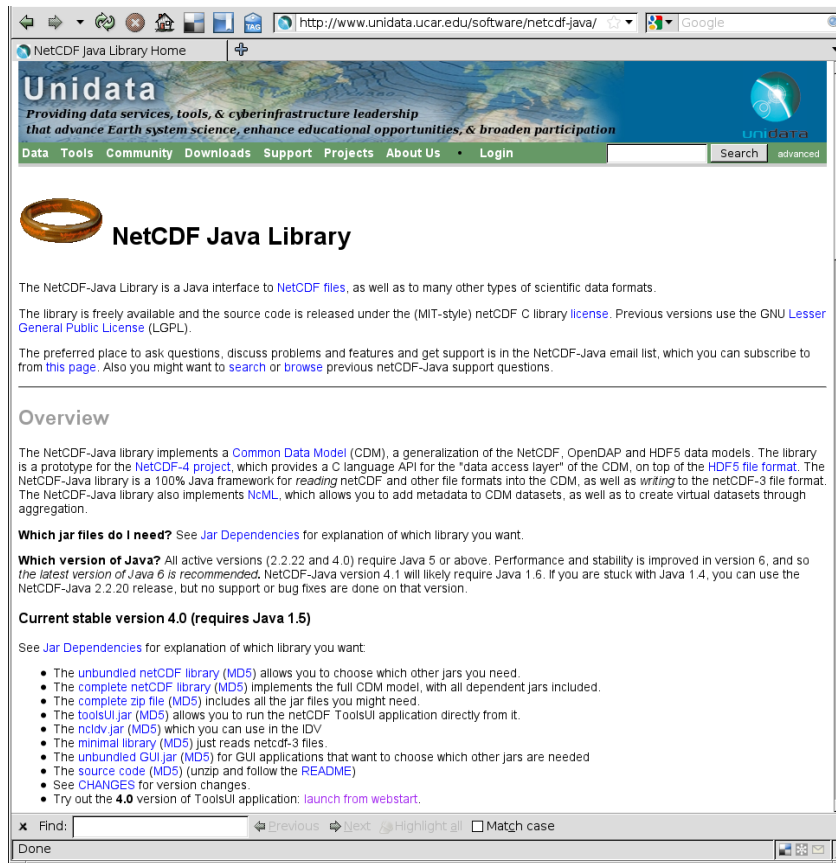
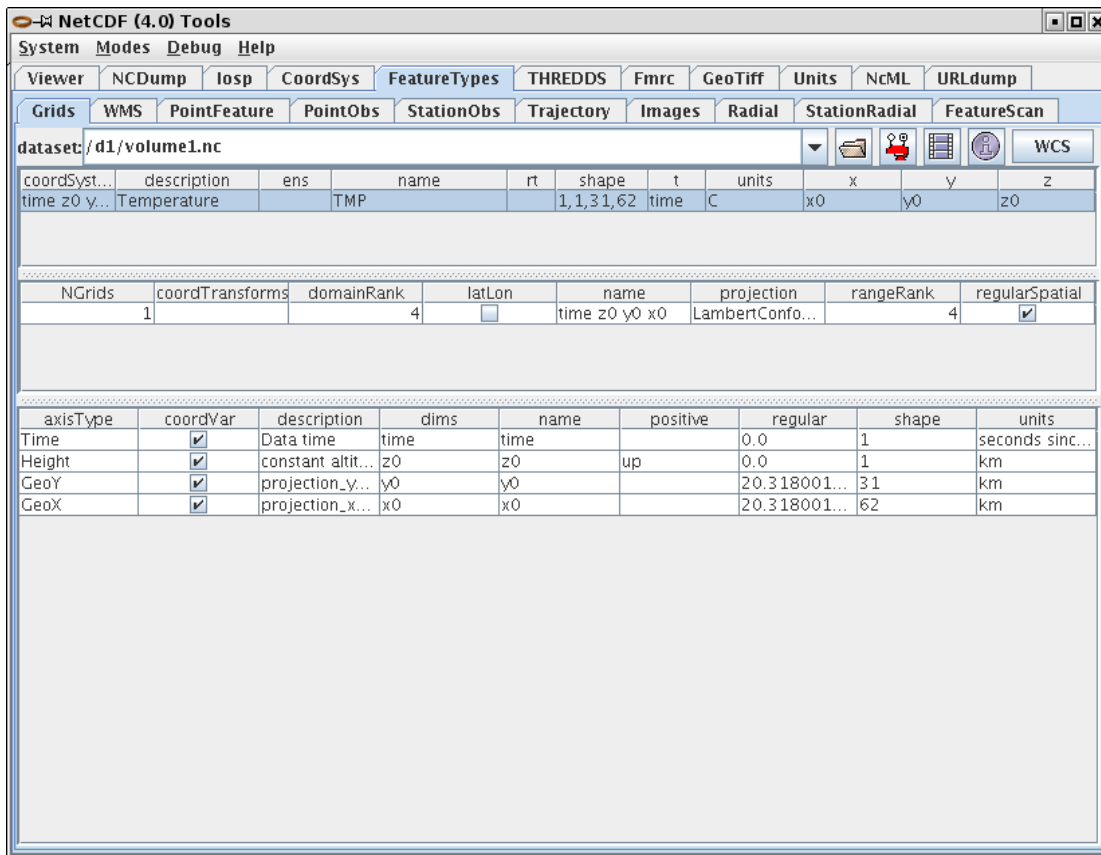


Figure 4.17: Unidata's Web Site



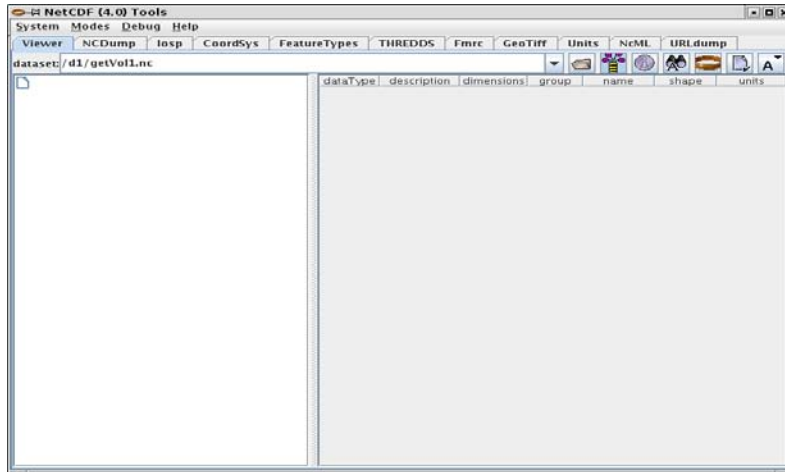


Figure 4.18: ToolsUI

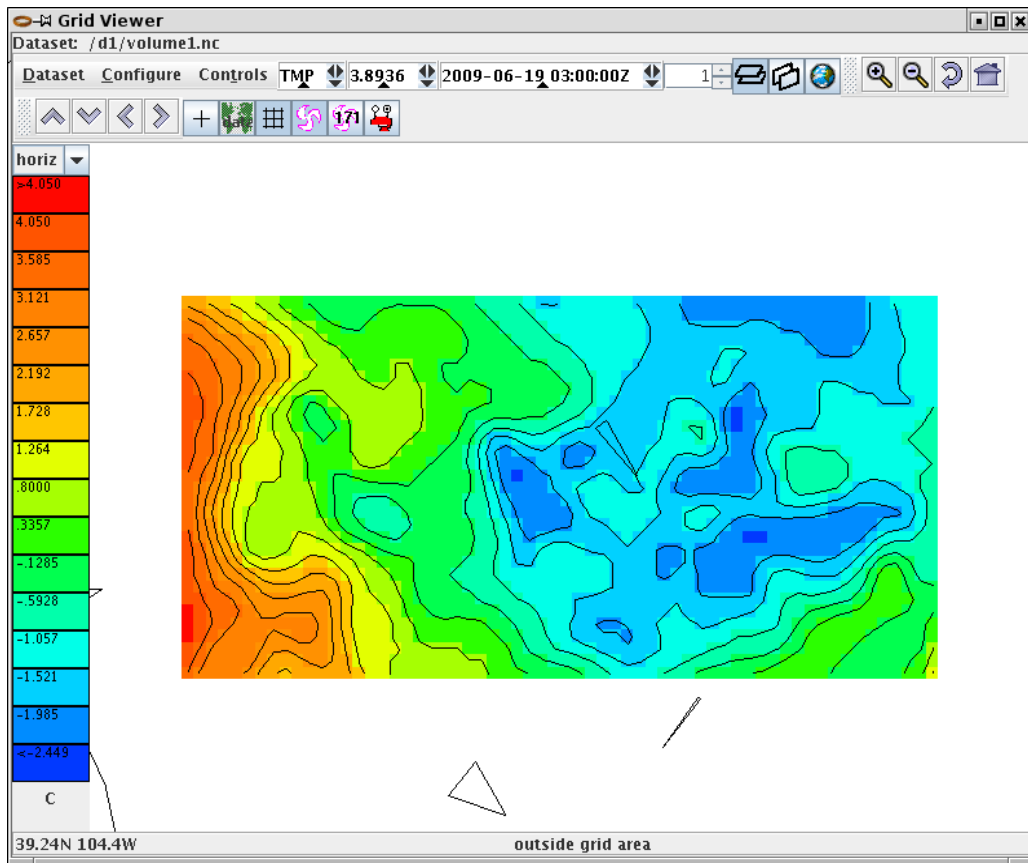




Figure 4.19: Grid Viewer

4.5.7 Executing GetCoverage for a Corridor

Returning to SoapUI, double click on the “**GetCorridor 3**” node, listed in the left-hand tree under the “**TestSuite for RUC (ruc-mdvConvert-nc4)**”, and maximize the new frame. Within the new frame, double click on the “**Test Request**” (i.e.,  Test Request) and maximize the new frame. Within that new frame, execute the test by pressing the “play” button (i.e., ). The test should execute successfully, and SoapUI should now look similar to Figure 4.20. The “**Test Request**” frame should be divided in half (or have two tabs), where the left hand side contains the XML for a GetCoverage request, and the right hand side contains the XML response (note that the SOAP headers have been hidden by SoapUI). Figure 4.20 depicts the GetCoverage request for a corridor of data. Note the following features of the request:

- The “**Identifier**” matches that of the test coverage dataset.
- The “**TemporalSubset**” for the request contains two valid times “**2009-06-19T03:00:00.000Z**” and “**2009-06-19T06:00:00.000Z**,” for which there is data available (according to the results of DescribeCoverage).
- The “**Identifier**” for the requested “**FieldSubset**” is “**TMP**”.
- Note the “**GridBaseCRS**” entry. It specifies the following:

1. 500 sample points for the corridor
2. The next several numbers specify waypoints for the corridor’s trajectory, in units of latitude, longitude, altitude in kilometers and time (seconds since 1970) respectively.

Waypoint 1: 39.863, -104.648, 1.609, 1245380400000 (i.e., 2009-06-19T03:00:00.000Z)

Waypoint 2: 40.795, -111.98, 1.288, 1245385800000 (i.e., 2009-06-19T04:30:00.000Z)

Waypoint 3: 47.451, -122.316, 0.132, 1245391200000 (i.e., 2009-06-19T06:00:00.000Z)

- Note the “**BoundingBox**” entry. This specifies the left/right and vertical dimensions for the corridor. The following significant values are specified:
 1. +/- 5.0 km extents in the lat/lon plane perpendicular to the trajectory.
 2. +/- 0.5 km extents in the vertical direction.
- Note the “**GridOffsets**” entry. This specifies the resolution in the left/right and vertical dimensions for the corridor. The following significant values are specified:
 1. 0.1 km resolution in the lat/lon plane perpendicular to the trajectory.
 2. 0.05 km resolution in the vertical direction.

Verify that the response contains the Identifier as listed in the request, and that the response frame looks similar to Figure 4.21. Click on the “**Attachments**” tab near the bottom of the response frame, and you should see one attachment as shown in Figure 4.21. This illustrates the successful application of MTOM

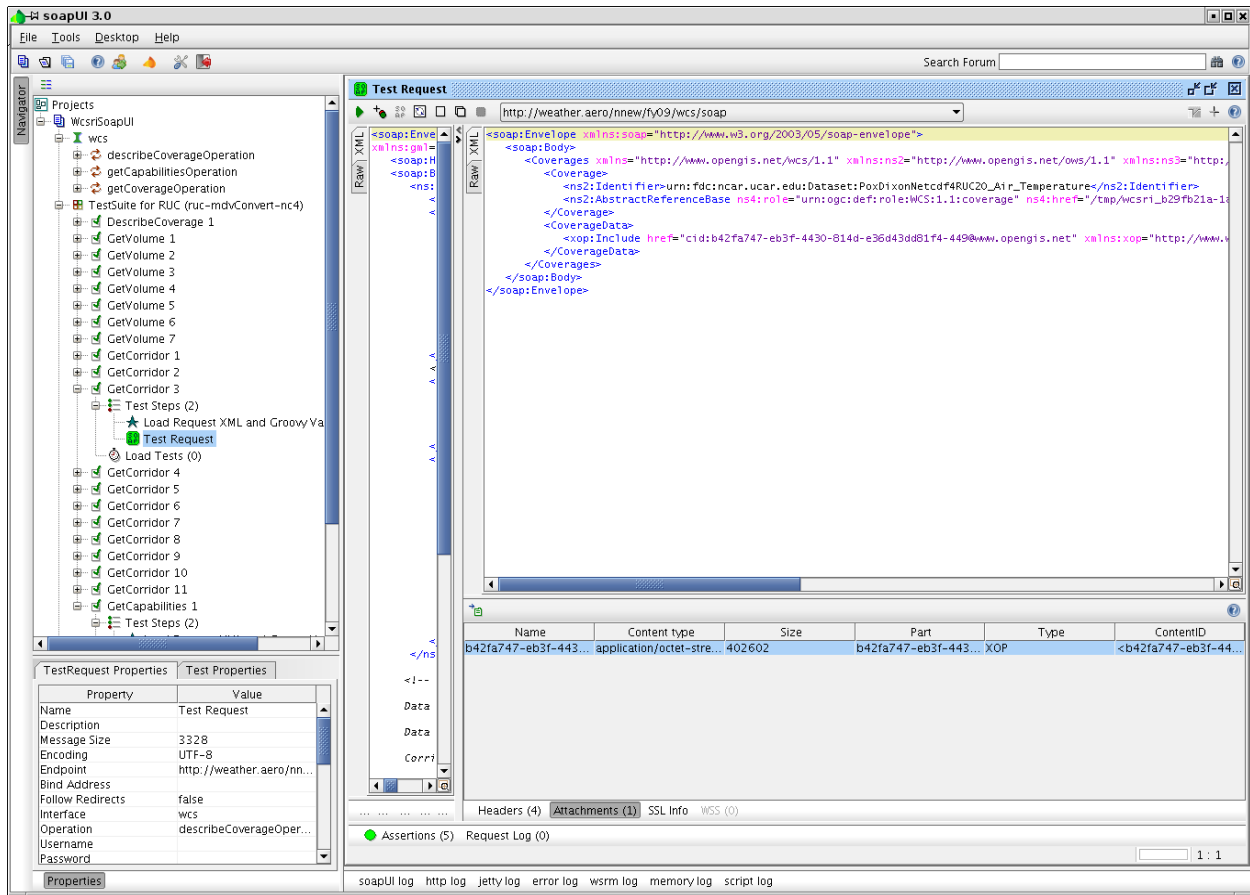


Figure 4.21: GetCorridor 3 Response

4.5.8 Visualizing GetCoverage Corridor Results

In this step, we will use a 3rd party application called `ncdump` to “visualize” the corridor obtained in the last test step. Since the corridor is 3-dimensional, it is difficult to visualize, and therefore we will just verify the *structure* of the corridor data file.

Use `ncdump` (version 4) to dump out the structure of the corridor obtained from the previous step, by typing the following from a command line:

```
ncdump corridor3.nc | more
```

The result should be similar to Figure 4.22. Glancing at the picture of a corridor displayed in Figure 4.23 may help in interpreting the results. In the `ncdump` output, notice that the resulting corridor has a z, y and x dimension. The x dimension has a value of 500, which is the number of sample points taken along the corridor’s trajectory. The y dimension is perpendicular to the corridor’s trajectory in the lat/lon plane, and contains $(5.0 \text{ km extent} + 5.0 \text{ km extent}) / 0.1 \text{ km resolution} + 1 = 101$ grid cells. The z dimension is altitude (km MSL) in the vertical, and contains $(0.5 \text{ km extent} + 0.5 \text{ km extent}) / 0.05 \text{ km resolution} + 1 = 21$ grid cells. Also note that the “time” variable is a function of x, the longitude and latitude variables are functions of y and x, and the altitude variable is a function of z and x. Lastly, note that TMP (i.e., Temperature) is a function of z, y and x. This test case illustrated the structure of Netcdf 4 corridor data returned as a result of issuing a GetCoverage request.

If you haven’t already, you may now close the SoapUI and ToolsUI applications.

```
Shell No. 2 - Konsole <2>
Session Edit View Bookmarks Settings Help

netcdf corridor3 {
dimensions:
  z_dim = 21 ;
  y_dim = 101 ;
  x_dim = 500 ;
variables:
  double time(x_dim) ;
  time:_FillValue = NaN ;
  time:_long_name = "time" ;
  time:_standard_name = "time" ;
  time:_units = "ms" ;
  float longitude(y_dim, x_dim) ;
  longitude:_FillValue = NaNf ;
  longitude:_long_name = "longitude" ;
  longitude:_standard_name = "longitude" ;
  longitude:_units = "degrees" ;
  float latitude(y_dim, x_dim) ;
  latitude:_FillValue = NaNf ;
  latitude:_long_name = "latitude" ;
  latitude:_standard_name = "latitude" ;
  latitude:_units = "degrees" ;
  float altitude(z_dim, x_dim) ;
  altitude:_FillValue = NaNf ;
  altitude:_long_name = "altitude" ;
  altitude:_standard_name = "altitude" ;
  altitude:_units = "km" ;
  float TMP(z_dim, y_dim, x_dim) ;
  TMP:_long_name = "Temperature" ;
  TMP:_standard_name = "TMP" ;
  TMP:_units = "C" ;
  TMP:_FillValue = NaNf ;

// global attributes:
  :_FileFormat = "Netcdf4" ;
  :_Conventions = "CF-1.4" ;
  :_history = "Produced by ModelDerive" ;
  :_source = "/d1/adds/system/data/ruc/mdv/Lambert/20090618/g_210000/f_00032400.mdv" ;
  :_title = "GFS model output." ;
  :_comment = ;

data:

time = 1245380400000, 1245380429924, 1245380459848, 1245380489772,
```

Figure 4.22: ncdump of corridor3.nc

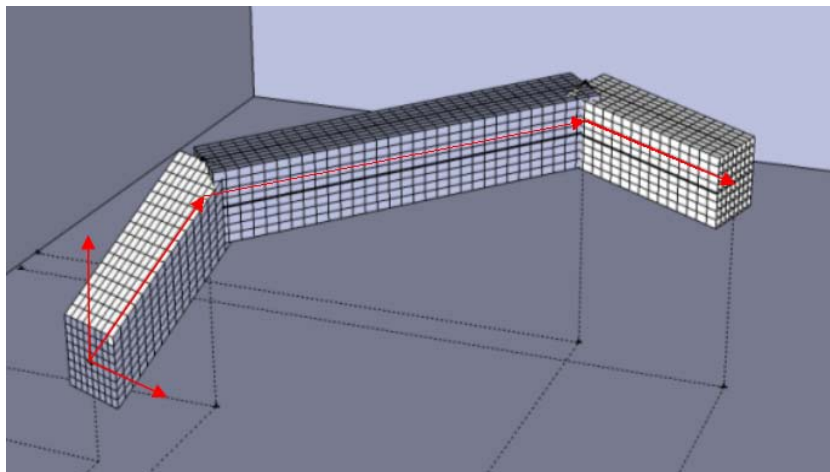


Figure 4.23: Sample Corridor

4.6 Verification with NNEW FY09 Integrated Java Application

The section will illustrate integration of coverages served by different WCSRI instances into a single integrated Java application. Launch the application by pointing a web browser to the following URL:

<http://weather.aero/nnew/fy09/demo/NNEWFY09Demo.jnlp>

or alternatively, visit:

<http://weather.aero/nnew/fy09/demo>

and click on the link labeled “**Launch the NNEW FY09 Demo.**”

The first time the application is started, it will take several moments for the application itself to download. Thereafter, the application will be cached on the client machine, and any updates to the application will be checked for and automatically downloaded each time the application is started.

Select and load the “**Default**” configuration in the “**Configuration Manager**” window, as shown in Figure 4.24.

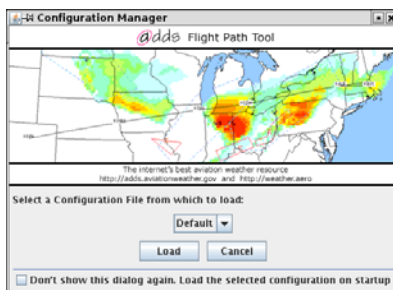


Figure 4.24: Configuration Manager

After a few moments, the main application window, titled “**NNEW IT Demonstration Client,**” will open. It should appear similar to Figure 4.25. At this point, the user will see a map of the continental United States, with an overlaid colored grid of temperature valid for a recent time at 16,000 ft.

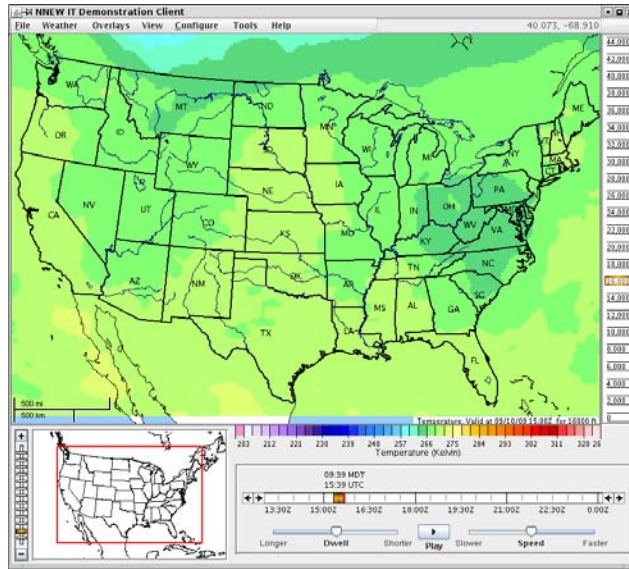


Figure 4.25: NNEW IT Demonstration Client

Select each gridded dataset in turn, by selecting it from the “**Weather**” heading in the menu bar. The gridded datasets are listed in Table 4.2 Gridded Data Sets **Error! Reference source not found.**, below. The datasets are served from different data server instances: RAL’s WCSRI 1.0 server, Lincoln Lab’s WCSRI 1.0 server, NOAA GSD’s WCSRI 1.0 server and NOAA NWS’s WCSRI 1.0 server. In addition, some of the datasets contain a forecast component and are 3-dimensional, whereas others are observational data with only 2 dimensions.

As appropriate for each dataset, use the time controller and the altitude controller to test the various datasets. Change the selected time by adjusting the orange slider (as in Figure 4.26) which will likely cause a data reload, depending on the server-side availability of data. Time loops can be done by using the animation controls on the bottom right-hand side of the screen. Animation is done by loading data, rendering images and caching the imagery on the first loop, and thereafter using the cached images for the animation – hence, the first loop in an animation is slower than subsequent loops. To test the 3-dimensionality of data, click on various altitudes within the altitude controller (as in Figure 4.27) which will also cause a data reload and re-rendering of the appropriate data “slice.” Notice that the bottom right-hand corner of the map will display a status message for all datasets that are visible. This includes the dataset’s valid time and altitude.

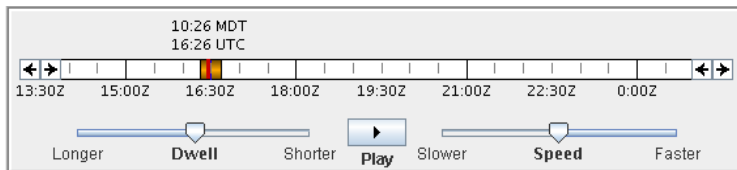


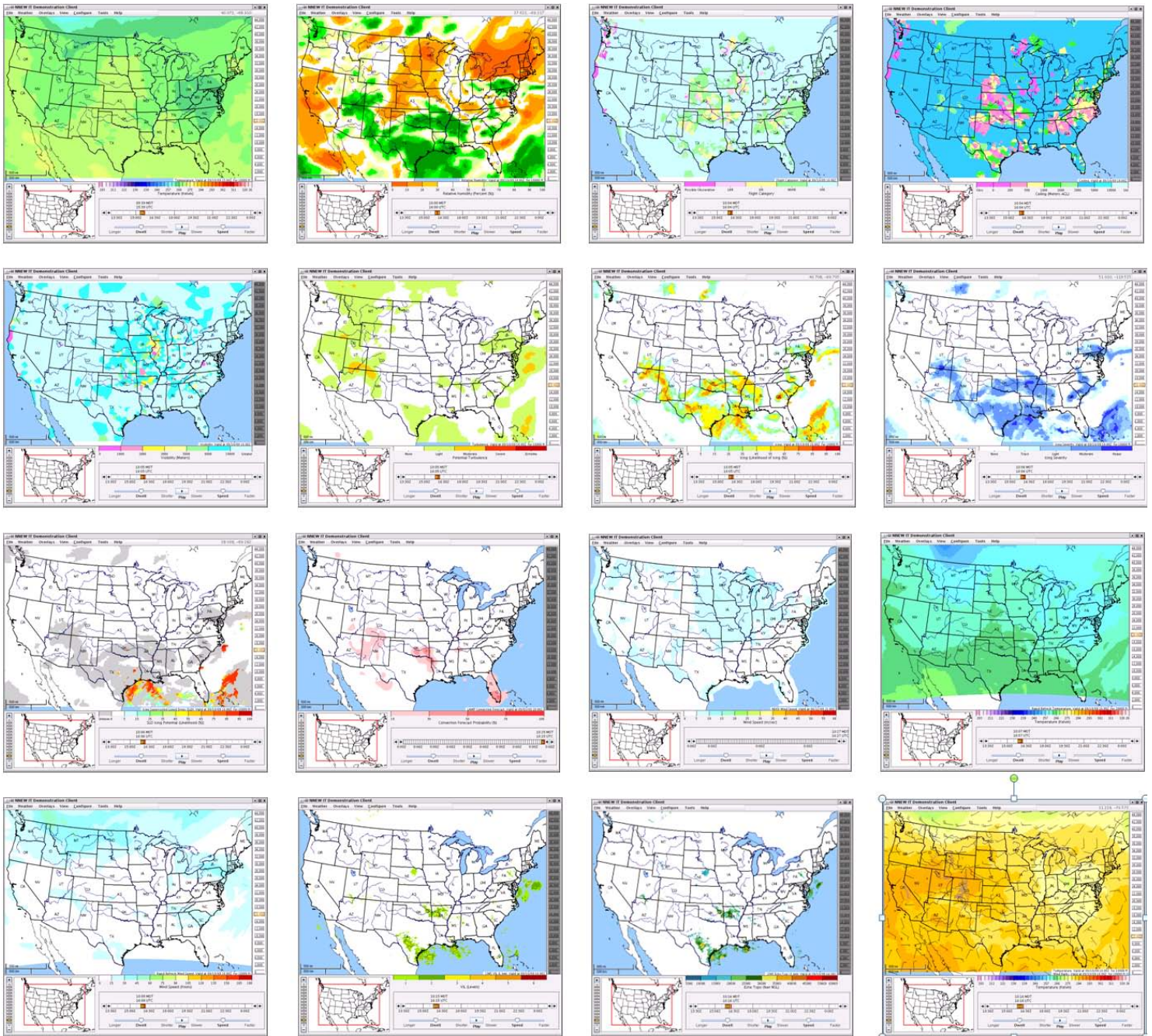
Figure 4.26: Time Controller



Figure 4.27: Altitude Controller

A visual catalog of each of the gridded products is shown below as a series of images. The order of the images is the same as the order of the datasets under the “**Weather**” menu. In general, the imagery displayed by the application for each dataset should be similar to the appropriate imagery shown in the visual catalog below. Clearly, the images displayed in the application will be different from those in the visual catalog due to differences between the selected time, altitude and zoom level. In addition, one might compare rendered data with plots from another source.

Table 4.1 Gridded Data Sets Visual Catalog



Zooming and panning should also be tested in the application, thereby testing re-retrieval of data based on geographic location. Zooming is done by double left-clicking at the desired location on the map to zoom in, and single right-clicking at the desired location to zoom out. Panning is accomplished by dragging the map with the left mouse button depressed. Each zoom or pan results in a new request for data by sending new latitude/longitude bounding box, altitude and time constraints to the server(s). For ease of use, a default geographic location can be selected from the “View” menu.

3-dimensional datasets as listed in **Error! Reference source not found.** should also be tested for cross-section (i.e., corridor) capabilities (with the exception of Rapid Refresh Wind Speed). From the “**Tools**” menu, select “**Flightpath**” → “**Click Way Points**” to enter the waypoints for a cross-section. For each waypoint, left click on the map. After you have created all of the waypoints, right click and select “**Show Cross-section**” to display the vertical cross-section window. Alternatively, submitting the last waypoint can be done with a double left click, also opening the cross-section window. Note that cross-sections are implemented essentially as GetCoverage requests for corridors of 0-length width, for a constant time and altitude.

Table 4.2 Gridded Data Sets

Dataset Name	Description	Host	Service	Protocol	Forecast	3D
Temperature	Rapid Update Cycle (RUC) model 20km	Exp ADDS	RAL WCSRI 1.0 Server	WCS 1.1.2 over SOAP	X	X
Relative Humidity	Rapid Update Cycle (RUC) model 20km	Exp ADDS	RAL WCSRI 1.0 Server	WCS 1.1.2 over SOAP	X	X
Wind Speed (i.e., Wind Barbs)	Rapid Update Cycle (RUC) model 20km	Exp ADDS	RAL WCSRI 1.0 Server	WCS 1.1.2 over SOAP	X	X
Flight Category	Ceiling and Visibility (C&V)	Exp ADDS	RAL WCSRI 1.0 Server	WCS 1.1.2 over SOAP	X	X
Ceiling	Ceiling and Visibility (C&V)	Exp ADDS	RAL WCSRI 1.0 Server	WCS 1.1.2 over SOAP	X	X
Visibility	Ceiling and Visibility (C&V)	Exp ADDS	RAL WCSRI 1.0 Server	WCS 1.1.2 over SOAP	X	X
Turbulence	Graphical Turbulence Guidance (GTG)	Exp ADDS	RAL WCSRI 1.0 Server	WCS 1.1.2 over SOAP	X	X
Icing	Current/Forecast Icing Potential (CIP/FIP)	Exp ADDS	RAL WCSRI 1.0 Server	WCS 1.1.2 over SOAP	X	X
Icing Severity	Current/Forecast Icing Potential (CIP/FIP)	Exp ADDS	RAL WCSRI 1.0 Server	WCS 1.1.2 over SOAP	X	X
Icing Super Cooled Liquid Droplets	Current/Forecast Icing Potential (CIP/FIP)	Exp ADDS	RAL WCSRI 1.0 Server	WCS 1.1.2 over SOAP	X	X
LAMP Convective	LAMP	NOAA/NWS	NOAA/NWS WCSRI 1.0	WCS 1.1.2 over SOAP	X	N/A

Forecast			Server			
NDFD Wind Speed	NDFD	NOAA/NWS	NOAA/NWS WCSRI 1.0 Server	WCS 1.1.2 over SOAP	N/A	N/A
Rapid Refresh Temperature	WRF Rapid Refresh, North America 13km model	NOAA/GSD	NOAA/GSD WCSRI 1.0 Server	WCS 1.1.2 over SOAP	X	X
Rapid Refresh Wind Speed	WRF Rapid Refresh, North America 13km model	NOAA/GSD	NOAA/GSD WCSRI 1.0 Server	WCS 1.1.2 over SOAP	X	X
CIWS VIL	CIWS Vertically Integrated Liquid	MIT/LL	MIT/LL WCSRI 1.0 Server	WCS 1.1.2 over SOAP	N/A	N/A
CIWS Echo Tops	CIWS Cloud Tops	MIT/LL	MIT/LL WCSRI 1.0 Server	WCS 1.1.2 over SOAP	N/A	N/A

5. Implementation Verification – Web Feature Service Reference Implementation (WFSRI)

Non gridded data includes, but is not limited to, airport, reporting station, and navaid information, surface and aloft observations, aircraft trajectories, and airspace volume boundaries. Standards for encoding and dissemination of these data are not as mature as those for gridded data. Ongoing work with Eurocontrol on the development of the Weather Exchange Model (WXXM, Version 1.1), together with standard interfaces for Web Feature Service (WFS, Version 1.1) by Open Geospatial Consortium (OGC) have laid the foundation for encoding and disseminating these data.

For the NNEW FY '09 IT Demonstration, the team has developed a Web Feature Service server that conforms to OGC WFS Specification 1.1 and provides features for publishing and retrieving WXXM Version 1.1 compliant feature data. The WFSRI supports both “pull” retrieval of data via a request/response OGC WFS compliant mechanism, as well as “push” retrieval of data via a subscription mechanism that extends the OGC WFS specification. Additionally, the WFSRI supports geospatial and temporal subsetting as well as filtered access to the data.

Figure 5.1 gives an architectural overview of the WFSRI. The WFSRI server Version 1.0 requires the following software be installed prior to the installation of the WFSRI itself:

- Oracle Database 11.1.0.6 with PatchSet 11.1.0.7. The Oracle installation must include:
 - Oracle Spatial
 - Oracle XDB
 - Oracle Advanced Queues
- Service Mix
- Apache Tomcat, version 6.X with JDK 1.6
- Java JDK 1.6

Oracle Configuration

To configure Oracle 11g for the WFSRI perform the following steps:

1. Download the WFSRI Oracle package from wxforge.

```
svn co
http://wxforge.wx.ll.mit.edu/svn/wfsri/tags/wfsri-1.0.0/src/sql
oracle-sql
```

2. The `oracle-sql` directory should contain the following sub-directories

- `install`
- `main`
- `test`
- `logs`
- `samples`

3. Install the WFSRI Package. Change directory to `oracle-sql/install` and type the following command on the command line:

```
sqlplus "sys/syspass as sysdba" @wfs_install <NNEW_WFS_USR>
      <password> <Tablespace-for-WFS> <Temp-Tablespace-for WFS>
```

Replace:

<syspass>	Password for your Oracle DBA
<NNEW_WFS_USR>	Name of the nnew_wfs_user
<password>	Password for the nnew_wfs_user. This password is also used as the default for the NNEW system user (NNEW_WFS).
<Tablespace-for WFS>	Name for the NNEW tablespace. This is the tablespace that holds the WFSRI metadata. While this can be set to use an existing tablespace, SYSAUX for example, we recommend using a distinct tablespace for the WFSRI metadata. The tablespace will be created by the installation script.
<Temp-Tablespace for WFS>	Name of tablespace to be used by the WFSRI for writing temporary files and logs.

This script creates a NNEW_WFS user, and installs all required WFSRI metadata tables and libraries into the NNEW_WFS user space. To test the installation of the NNEW_WFS user, log into Oracle as the NNEW_WFS user, and type the following command:

```
sqlplus NNEW_WFS/<password>
sqlplus> select table_name from user_tables;
```

You should see the following tables:

```
WFS_PRODUCER
WFS_PROD_FEATURETYPE
WFS_FEATURETYPE_QUEUE
WFS_ADMINFEATURE
ERRLOG
LOGGER
```

WFSRI Software Installation

To install the WFSRI, download the WFSRI Version 1.0 (wfsri_version1.0-beta2.war) war file from <https://wiki.ucar.edu/display/NNEW/Releases> and perform the following steps:

1. Drop the downloaded war file to the Apache Tomcat webapps directory.
2. Re-start Tomcat to build the war (soap-1.0.0) directory under the webapps directory, hereby referenced as \$WFSRI_HOME.

3. Download and modify the `properties` files to reflect the configuration for the WFSRI. Checkout the `conf` directory from the FUSE-WFSRI project:

```
svn checkout
```

```
http://wxforge.wx.ll.mit.edu/svn/fuse-wfsri/tags/soap-1.0.0/conf conf
```

Edit the `wfsri.properties` file to set the properties of the WFSRI.

- `brokerURLActivemq` – The URL to the ActiveMQ broker.
- `oracle.dataSource.url` – The URL to the Oracle data source.
- `oracle.dataSource.username` – The username of the Oracle data source. This should be set to `nnew_wfs`.
- `oracle.dataSource.password` – The password for the `NNEW_WFS` user.
- `wfs.url` – The URL for the service.

Set the environment variable `CONFIG_DIR` to the location of the `conf`.

```
setenv CONFIG_DIR ../conf/
```

4. Security Configuration. Version 1.0 of the WFSRI is not currently integrated with an LDAP service for authenticating the username/password combinations. In lieu of the LDAP service, to add a username/password pair for the service edit the `CamelContext.xml` file in `$WFSRI_HOME/WEB-INF`. Edit the property `userMap` in the `memoryUserDetailsService` bean in `CamelContext.xml` to list the username/password pairs. The format for this is:

```
username=password,ROLE_USER
```

For example, a user `tom` with a password of `riddle` would be listed as:

```
<bean id="memoryUserDetailsService"
  class="org.springframework.security.userdetails.memory.InMemoryDaoImpl">
  <property name="userMap">
    <value>tom=riddle,ROLE_USER</value>
  </property>
</bean>
```

5. Re-start Tomcat to capture the latest changes in the `properties` files.

The remainder of this test plan assumes that the WFSRI installed at MIT Lincoln Laboratory with the endpoint: <http://ngen-wfsri.wx.ll.mit.edu/soap/wfs> will be utilized.

The following series of tests demonstrate the functionality of the Web Feature Service Reference Implementation (WFSRI) using a set of clients. The sequence of the tests below follows the lifecycle of the WFSRI from set up to publishing feature data to feature retrieval. Figure 5.1 highlights the primary flow of tasks that are undertaken by a feature producer and a feature consumer. Lightning and MDCR data will be used throughout this demonstration. The Lightning data will be displayed using the Google Earth Subscription Client. Client side adaptation of the CIWS display will be demonstrated as well.

Figure 5.2 highlights the CIWS and WFSRI interaction. Screenshots are provided with which to verify the validity of the output.

Section 5.1 describes the setup of the test environment, that is, the *test clients* used throughout this demonstration. The remainder of the sections demonstrates the functionality highlighted in Figure 5.1.

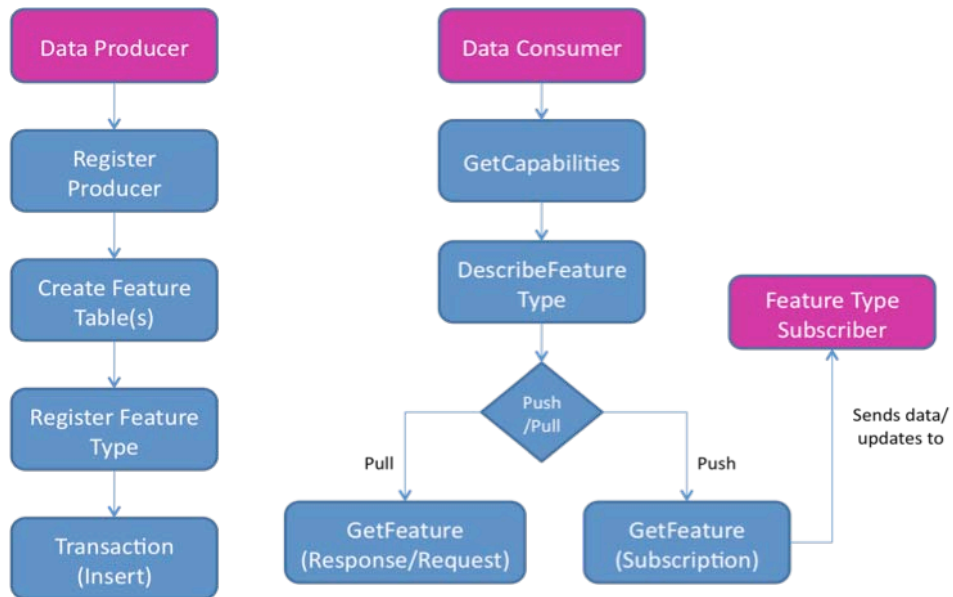


Figure 5.1 WFSRI Flow

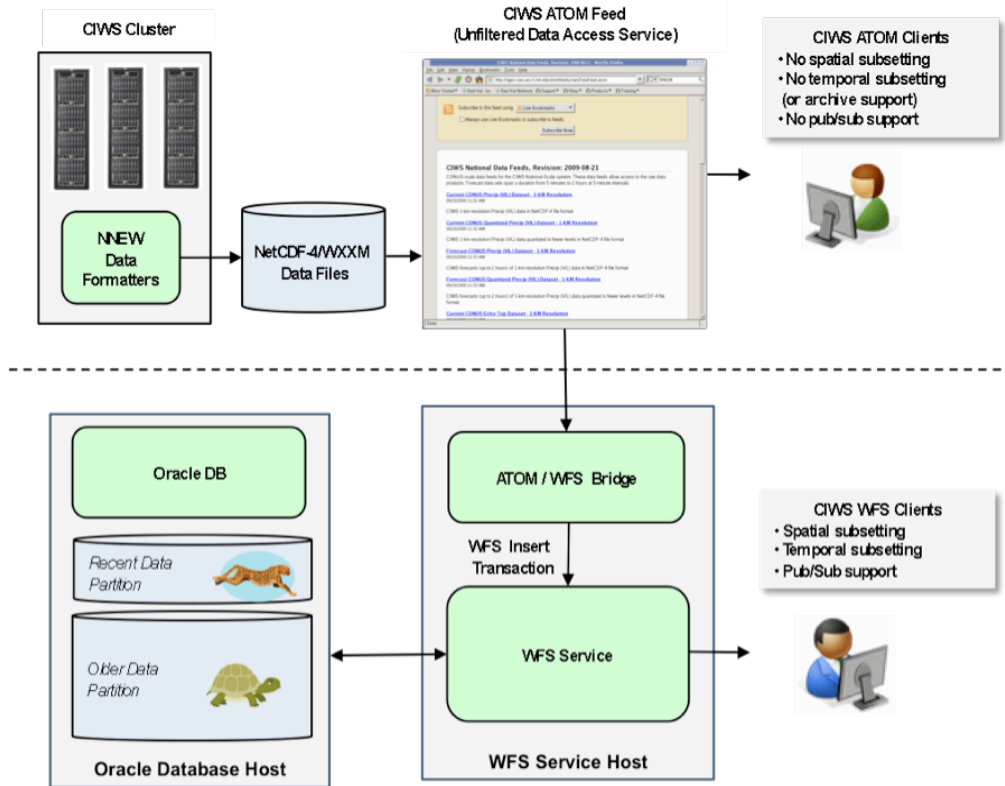


Figure 5.2: CIWS and WFSRI Interaction

5.1 Test Environment and Setup

5.1.1 Administration Client

The WFSRI Administrator provides administrative access, from registering a publisher/producer, to registering feature types, to monitoring the WFSRI state. The WFS Administration client is bundled with the WFSRI and can be accessed at:

<http://ngen-atomwfs-data.wx.ll.mit.edu:61618/WFSAdmin>

5.1.2 Publish Clients

The Publish Clients allow data providers to publish their feature data to the WFSRI server. The WFSRI Version 1.0 provides two sample publish clients: a general purpose Publish client and a continuous Lightning Publisher that integrates with and publishes the CIWS Lightning ATOM feed data to the WFSRI server hosted at MIT Lincoln Laboratory. Figure 5.2: CIWS and WFSRI Interaction, shows a schematic of the interaction between the CIWS ATOM feed, the ATOM-WFS-Bridge and the WFSRI.

Generic Client

Requirements:

- Maven 2.0.9
- JDK 1.6

Installation:

To install the generic client follow the steps outlined below.

1. Download the client from <http://wxforge.wx.ll.mit.edu>

```
svn co
http://wxforge.wx.ll.mit.edu/svn/fuse-wfsri/trunk/client/soapclient
genericClient
```
2. Change directory to `genericClient`, and compile the client.

```
mvn compile; mvn appassembler:assemble
```
3. The above command creates an executable `genericClient/target/bin/SoapClient`.
 Ensure that the `SoapClient` is executable.

```
chmod u+x target/bin/SoapClient
```
4. Set the REPO environment variable to point to your maven repository.

```
REPO=~/.m2/repository
export REPO
```

Secure Publish Client

Requirements:

- Maven 2.0.9
- JDK 1.6

Installation:

1. Build the Client Library: To build the client library: Checkout the `ogc-bindings` project from <http://wxforge.wx.ll.mit.edu>

```
svn checkout
http://wxforge.wx.ll.mit.edu/svn/ogc-bindings/tags/ogc-bindings-1.0
ogc-bindings
```
2. Navigate to the `ogc-bindings/xmlbeans/nawx/1.1.0_gml311` directory, then run the command

```
mvn install
```
3. Check out the client library from wxforge.wx.ll.mit.edu

```
svn checkout
http://wxforge.wx.ll.mit.edu/svn/new/ri-client/tags/wfsri-client-1.1
ri-client
```

4. Navigate to the ri-client/trunk/wfsri directory in the ri-client project, run the command

```
mvn install
```



```
mvn appassembler:assemble
```
5. This will generate the SamplePublisherClientApp in target/bin. Add execution privileges to the SamplePublisherClientApp.

```
chmod 755 target/bin/*
```

5.1.3 Retrieval Clients

The Retrieval Clients allow data consumers to access filtered feature data from the WFSRI server. The WFSRI Version 1.0 provides two sample retrieval clients: a command line Request/Response api through the generic client used for publishing data and an integrated Google Earth subscription client.

Google Earth Subscription Client

Requirements:

The Google Earth Subscription client requires that Google Earth be installed prior to its installation. To install Google Earth:

1. Download Google Earth from:

<http://earth.google.com/download-earth.html>
2. Make the downloaded Google Earth file executable

```
chmod u+x GoogleEarthLinux.bin
```
3. Install Google Earth by executing this file by typing `./GoogleEarthLinux.bin` in the directory where the file was downloaded. The install process will attempt to create a directory `$HOME/google-earth`. It is best if this directory is mounted on a local disk. If this is not a local disk, have the install write to a local disk. Go to the directory where Google Earth was installed and run it by typing:

```
./googleearth
```

Installation:

To install the Integrated Subscription Client:

1. Checkout the `wfs-transformer` project from <http://wxforge.wx.ll.mit.edu>

```
svn co
http://wxforge.wx.ll.mit.edu/svn/new/ri-client/trunk/wfs-transformer
```
2. Change directory to `wfs-transformer`. Compile the `wfs-transformer` code.

```
cd wfs-transformer
mvn compile ; mvn appassembler:assemble
```

This will create the directory `target/bin` and in there a shell script called `GMLToKmlTest`.
Make this script executable by typing:

```
chmod +x target/bin/GMLToKmlTest
```

3. Now set the `REPO` environment variable to your maven repository.

```
setenv REPO $HOME/.m2/repository
```

For `csh` style shells, or for `bash` shells type:

```
REPO=$HOME/.m2/repository ; export REPO
```

4. Next create a directory for the `kml` output to go, and populate it with some default files.

```
mkdir $HOME/kml
cp wfs-transformer/src/main/resources/lightning* $HOME/kml
```

Change directory to `$HOME/kml` and edit the file `LightningFile.kml`. Replace the `ll/wlp/km` with the path `$HOME/kml`, with the environment variable expanded. Take care to maintain the double slash after 'file:'

For example, with an FAA user's home directory:

```
<href>file:///home/faauser/kml/lightning.kml</href>
```

5.1.4 CIWS Display

Requirements:

- JDK 1.6

Installation:

1. Insert the CIWS NNEW Display software disk into a CD or DVD ROM drive.
2. Make a directory on a disk with write privileges and at least a gigabyte of space.
3. Open two terminals and change both to the newly created directory.
4. In the first terminal, run the following command:

```
tar -xvf PATH_TO_CDRROM/ciws_nnew_display.tar
```

5. In the first terminal edit the “run” and the “run_LightningProcessor” scripts so that you can specify your http proxy settings. There is a section for the proxy settings near the top of the file.

For no proxy use:

```
PROXYHOST=" "
```

```
PROXYPORT=" "
```

If you do have a proxyhost (substitute somehost... for your proxy host):

```
PROXYHOST="-Dhttp.proxyHost=somehost.somedomain.org"
```

```
PROXYPORT="-Dhttp.proxyPort=8080"
```

6. In the first terminal execute the script by typing:
./run
7. In the second terminal run the “run_LightningProcessor” script
./run_LightningProcessor
8. To exit the display, invoke the “**System**” → “**Exit**” menu item. Type **Ctrl-c** from the second terminal to terminate the lightning processor.

Notes:

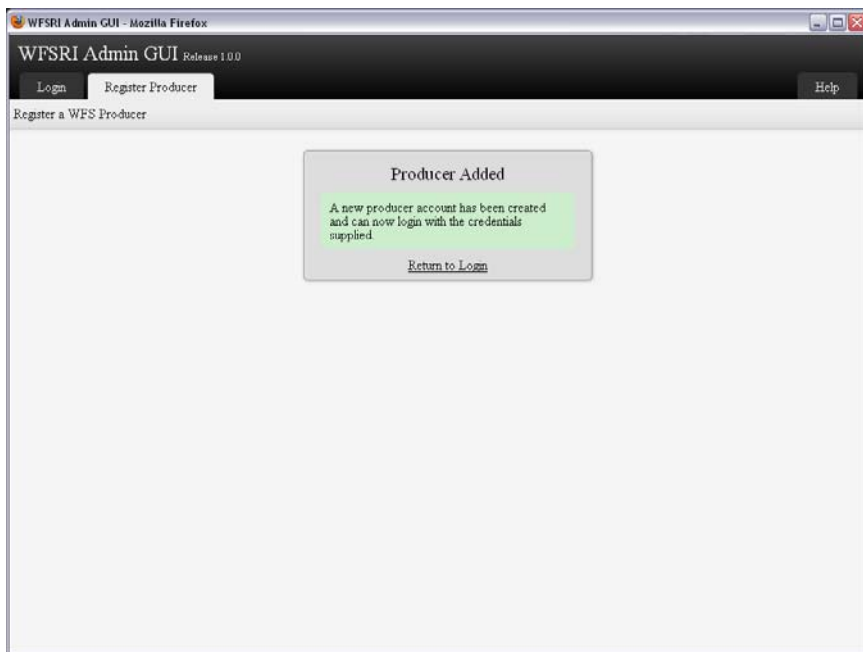
Currently the program downloads and keeps product history. If disk space is an issue, stop the display and delete the contents of the “data” and “tmp” directories (leave the “data” and “tmp” directories).

5.2 Register New Producer Using the WFSRI Administrator

Launch the WFSRI Administrator by pointing a web browser to the following URL:

<http://ngen-atomwfs-data.wx.ll.mit.edu:61618/WFSAdmin>

Click on the “**Register a new Producer**” link at the bottom of the login screen. Fill in a name and password and click “**Register Producer.**”



You should see a screen that indicates that the producer has been successfully created.

5.3 Create Feature Table

MIT/LL staff will use a VPN connection to a MIT/LL machine to complete step 5.3

Prior to registering a feature type that will be served up using the WFSRI, the producer must create the table (Feature Type Table) that will store the feature instance data. A default “empty” schema is created during the registration of the producer. To create the feature table for MDCR data:

1. Change to the AIREP directory (cd AIREP)
2. Login to the Oracle database using sqlplus and the producer credentials created above.

```
sqlplus AIREP/AIREP
```

3. You should see a sqlplus prompt. Type the sql command shown below:

```
sql> select table_name from user_tables;
```

4. This statement should return “No Rows Found”. To create the AIREP feature table, execute the following from the sql prompt:

```
sql> @createAIREPTable.sql
```

5. You should see the following output:

```
SQL> @createAIREPTable.sql
```

```
Table created.
```

```
Sequence created.
```

```
Trigger created.
```

```
Trigger altered.
```

```
1 row created
```

```
Index created.
```

```
Grant succeeded.
```

5.4 Register Feature Type Using the WFSRI Administrator

Check out the AIREP table script from wxforge.wx.ll.mit.edu

```
svn co http://wxforge.wx.ll.mit.edu/svn/wfsri/trunk/examples/AIREP  
AIREP
```

To register a feature type login with the credentials of the producer registered in Section 5.2. Click on the “**Register Feature Type**” tab.

Fill in the fields on the form as shown below. Then, use the “**Browse**” buttons to find the “**Feature Type Description**” and “**Describe Feature Type**” files.

WFSRI Admin GUI - Mozilla Firefox

WFSRI Admin GUI Release 1.0.0 Welcome AIRREP Logout

Feature Types Register Feature Type Schemas Cache Help

Register a new Feature Type

Feature Type Information

Feature Type Name: AircraftReport

Feature Type URL: http://www.eurocontrol.int/wfs/1.1

Feature Type Alias: trmys

Feature Member NS:

Feature Member Name:

Feature Type Description:

Use a previously uploaded file or Upload a file

Select a previously uploaded file or trmys/AircraftReport.xml Browse

Describe Feature Type:

Use a previously uploaded file or Upload a file

Select a previously uploaded file or trmys/AircraftReport.xsd Browse

If you upload a file with the same name as one in the cache, the copy in the cache will be overwritten.

Schema Information

Schema Location:

Schema Name: Please select a schema... Select Schema

Use the drop-down and the “**Select Schema**” button to choose the Feature Table. Select one or more columns in the “**Column Info**” and “**Mandatory Column Info**” sections.

WFSRI Admin GUI - Mozilla Firefox

WFSRI Admin GUI Release 1.0.0 Welcome AIRREP Logout

Feature Types Register Feature Type Schemas Cache Help

Register a new Feature Type

Schema Information

Schema Location:

Schema Name: AIRREP Select Schema

Selecting a schema will automatically fill in the table, primary key column and primary spatial column.

Table Name: AIRCRAI Select Primary Key Column: ID Primary Spatial Column: LOCATION

Column Info: Mandatory Column Info:

ID ID

MESSAGE MESSAGE

SYS_NC00003\$ SYS_NC00003\$

LOCATION LOCATION

SYS_NC00005\$ SYS_NC00005\$

SYS_NC00006\$ SYS_NC00006\$

SYS_NC00007\$ SYS_NC00007\$

SYS_NC00008\$ SYS_NC00008\$

SYS_NC00009\$ SYS_NC00009\$

SYS_NC00010\$ SYS_NC00010\$

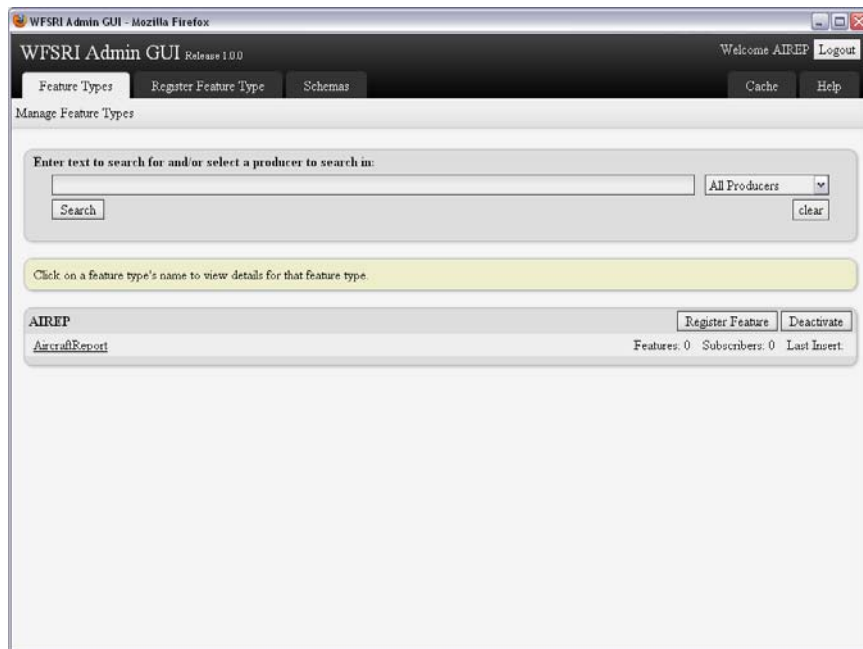
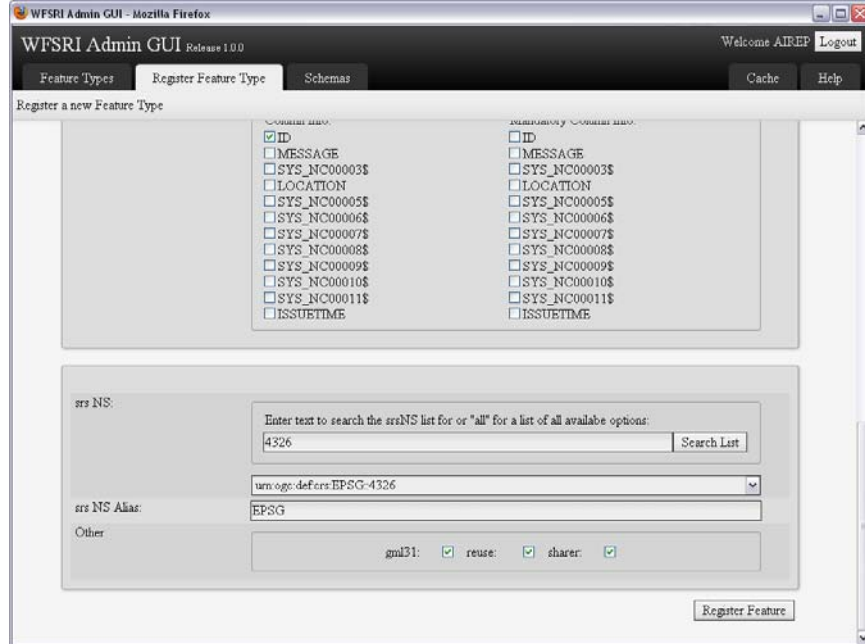
SYS_NC00011\$ SYS_NC00011\$

ISSUETIME ISSUETIME

Make sure ID is checked for column and mandatory column.

Use the text box and “**Search List**” button to retrieve supported Spatial Reference System namespaces.

Click the “**Register Feature**” button to register the Feature Type.



The “**Manage Feature Types**” tab should be presented to the Producer showing that the Feature has been registered successfully. (Leave this window available)

5.5 Transaction Insert

5.5.1 Using the Generic Client

The WFS-T specification provides interfaces to insert, update and delete feature data from the specified WFS server. The WFSRI supports only transaction insert. To insert MDCR feature data into the MIT LL WFS server:

1. Download the MDCR data from wxforge.wx.ll.mit.edu. If you have already downloaded the AIREP directory in Section 5.3, then skip this step. Else,

```
svn co http://wxforge.wx.ll.mit.edu/svn/wfsri/trunk/examples sample-requests
```

The AIREP directory contains several MDCR data files, labeled MDCR1...n.xml. Each file represents a MDCR feature wrapped in an OGC WFS transaction element.

Type

```
cat ~/wxforge/sample-requests/AIREPTestData/mdcr.xml
```

```
<?xml version="1.0" encoding="UTF-8"?>
<wfs:Transaction service="WFS" version="1.1.0"
  xmlns:wfs="http://www.opengis.net/wfs"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.opengis.net/wfs http://schemas.opengis.net/wfs/1.1.0/WFS-transaction.xsd">
  <wfs:Insert>

<avwx:AircraftReport
  xmlns:avwx="http://www.eurocontrol.int/wxxs/1.1"
  xmlns:wx="http://www.eurocontrol.int/wx/1.1"
  xmlns:wxont="http://wmo.int/ontologies/wx.owl#"
  xmlns:om="http://www.opengis.net/om/1.0/gml32"
  xmlns:gml="http://www.opengis.net/gml"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  gml:id="id0"
  xsi:schemaLocation="http://www.eurocontrol.int/wxxs/1.1 .int/eurocontrol/wxxs/1.1.0/wxxs.xsd">
<!--<Example of a MDCR based upon exmample ADMAR-->
<avwx:aircraftReportType>AIREP</avwx:aircraftReportType>
<avwx:aircraftId codeSpace="urn:icao:Aircraft:type">IUAX02</avwx:aircraftId>
<avwx:airspaceWxObservation>
<wx:Observation gml:id="id6">
<om:samplingTime>
<gml:TimeInstant gml:id="id8">
  <gml:timePosition>2009-08-28T03:19:00Z</gml:timePosition>
</gml:TimeInstant>
</om:samplingTime>
<om:procedure xlink:href="urn:fdc:icao:procedure:AircraftReport"></om:procedure>
<om:observedProperty
  xlink:href="http://www.eurocontrol.int/ont/avwx/1.1/wx.owl#AirspaceWx"></om:observedProperty>
<om:featureOfInterest>
<avwx:Airspace gml:id="id4">
<gml:location>
<gml:Point gml:id="id5" srsName="urn:ogc:def:crs:EPSG::4326">
<gml:pos>39.49 -78.25 6084.0 </gml:pos>
</gml:Point>
</gml:location>
</avwx:Airspace>
</om:featureOfInterest>
<om:result>
<avwx:AirspaceWx gml:id="id10">
```

```

<avwx:windSpeed uom="kt">8</avwx:windSpeed>
<avwx:windDirection uom="deg">229</avwx:windDirection>
<avwx:airTemperature uom="C">262</avwx:airTemperature>
<avwx:turbulence>
<avwx:Turbulence gml:id="tb1"></avwx:Turbulence>
</avwx:turbulence>
</avwx:AirspaceWx>
</om:result>
</wx:Observation>
</avwx:airspaceWxObservation>
</avwx:AircraftReport>
</wfs:Insert>
</wfs:Transaction>

```

2. Change directory to where you downloaded the generic client (see Step 5.1).

```
cd genericClient
```

3. Execute the generic client as follows:

```
target/bin/SoapClient http://ngen-wfsri.wx.ll.mit.edu/soap/wfs
../sample-requests/AIREPTestData/mdcr.xml AIREP AIREP
```

4. Verify that the following message is returned from the client:

```

Sep 11, 2009 5:45:50 PM org.springframework.ws.soap.saaj.SaajSoapMessageFactory afterPropertiesSet
INFO: Creating SAAJ 1.3 MessageFactory with SOAP 1.1 Protocol
<?xml version="1.0" encoding="UTF-8"?><xml-fragment><wfs:TransactionSummary
xmlns:wfs="http://www.opengis.net/wfs"><wfs:totalInserted>1</wfs:totalInserted></wfs:TransactionSummary><
wfs:InsertResults xmlns:wfs="http://www.opengis.net/wfs"><wfs:Feature handle=""><ogc:FeatureId
xmlns:ogc="http://www.opengis.net/ogc" fid="1"/></wfs:Feature></wfs:InsertResults></xml-fragment>

```

5. Refresh the WFSRI Admin GUI and verify features has incremented to 1

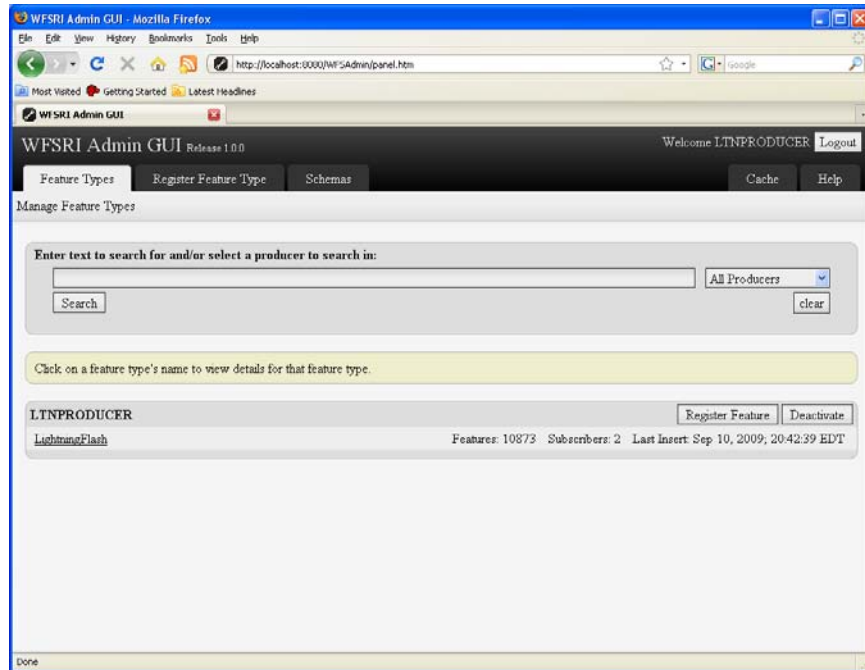
5.5.2 Using the ATOM-WFS Bridge

The ATOM-WFS bridge can be used to publish data to a WFS server. To publish data to the WFS server using the ATOM-WFS bridge, change to the directory where ATOM-WFS-Bridge was downloaded and installed. Start the ATOM-WFS-Bridge using the following command:

```
mvn jetty:run
```

This connects to the CIWS ATOM feed and publishes data to the WFS Server.

Verify that data is being published to the MIT LL WFS Server by logging into the WFS Admin using the username LTNPRODUCER and the password LTN. Verify that you see a number greater than 0 for the features under the “**LightningFlash**” feature type.



5.6 GetCapabilities Using the Generic Client

To retrieve the capabilities offered by the WFS server, you can pass in a GetCapabilities request to the generic client used in Section 5.4 for publishing data. The WFSRI comes bundled with a set of examples including a sample GetCapabilities request. To download the examples, checkout the samples directory from wxforge using :

```
svn co
http://wxforge.wx.ll.mit.edu/svn/wfsri/trunk/examples sample-requests
```

Change directory to where the generic client was downloaded and installed. Run the following command:

```
target/bin/SoapClient http://ngen-wfsri.wx.ll.mit.edu/soap/wfs
../sample-requests/getCapabilities.xml CONSUMER1 CONS
```

Verify that the output returned is similar to the text below (*only a snapshot shown*).

```
<ogc:Filter_Capabilities>
<ogc:Spatial_Capabilities>
<ogc:GeometryOperands>
<ogc:GeometryOperand>gml:Envelope</ogc:GeometryOperand>
<ogc:GeometryOperand>gml:Point</ogc:GeometryOperand>
<ogc:GeometryOperand>gml:LineString</ogc:GeometryOperand>
<ogc:GeometryOperand>gml:Polygon</ogc:GeometryOperand>
<ogc:GeometryOperand>gml:ArcByCenterPoint</ogc:GeometryOperand>
<ogc:GeometryOperand>gml:CircleByCenterPoint</ogc:GeometryOperand>
```

```

<ogc:GeometryOperand>gml:Arc</ogc:GeometryOperand>
<ogc:GeometryOperand>gml:Circle</ogc:GeometryOperand>
<ogc:GeometryOperand>gml:ArcByBulge</ogc:GeometryOperand>
<ogc:GeometryOperand>gml:Bezier</ogc:GeometryOperand>
<ogc:GeometryOperand>gml:Clothoid</ogc:GeometryOperand>
<ogc:GeometryOperand>gml:CubicSpline</ogc:GeometryOperand>
<ogc:GeometryOperand>gml:Geodesic</ogc:GeometryOperand>
<ogc:GeometryOperand>gml:OffsetCurve</ogc:GeometryOperand>
<ogc:GeometryOperand>gml:Triangle</ogc:GeometryOperand>
</ogc:Filter_Capabilities>
</wfs:WFS_Capabilities>

```

5.7 DescribeFeatureType Using the Generic Client

The DescribeFeatureType request returns the XML Schema descriptions of the one or more specified Feature types. To execute a DescribeFeatureType request, run the following command:

```

target/bin/SoapClient http://ngen-wfsri.wx.ll.mit.edu/soap/wfs
../sample-requests/describeFeatureType.xml CONSUMER1 CONS

```

Verify that this command returns the wxLightning.xsd.

INFO: Creating SAAJ 1.3 MessageFactory with SOAP 1.1 Protocol

```

<?xml version="1.0" encoding="UTF-8"?><wfs:DescribeFeatureTypeResponse xmlns:wfs="http://www.opengis.net/wfs-
util"><schema xmlns="http://www.w3.org/2001/XMLSchema" xmlns:gml="http://www.opengis.net/gml"
xmlns:nawx="http://www.faa.gov/nawx/1.1" xmlns:wx="http://www.eurocontrol.int/wx/1.1"
attributeFormDefault="unqualified" elementFormDefault="qualified" targetNamespace="http://www.faa.gov/nawx/1.1"
version="1.0">
<annotation>
<documentation>
Wx schema file for encoding lightning flash data
</documentation>
</annotation>
<import namespace="http://www.opengis.net/gml" schemaLocation="../../../../net/opengis/gml/3.1.1/base/gml.xsd"/>
<import namespace="http://www.eurocontrol.int/wx/1.1"
schemaLocation="../../../../int/eurocontrol/wx/1.1.0_gml311/wx.xsd"/>

<element name="LightningFlash" substitutionGroup="gml:_Feature" type="nawx:LightningFlashType"/>
<complexType name="LightningFlashType">
<annotation>
<documentation>
The LightningFlashType is an extension of wx:AbstractWxFeatureType to allow
encoding of lightning flash data.
</documentation>
</annotation>
<complexContent>
<extension base="wx:AbstractWxFeatureType">

```

```

<sequence>
  <element minOccurs="0" name="associatedFeatureID" type="ID">
    <annotation>
      <documentation>
        Optional associated feature ID for this lightning flash. For
        example, this could be the gml:id of a StormCell that the
        lightning flash is associated with.
      </documentation>
    </annotation>
  </element>
  <element name="strength" type="gml:MeasureType">
    <annotation>
      <documentation>
        Strength and polarity, typically expressed as a signed value
        in kiloamps.
      </documentation>
    </annotation>
  </element>
  <element name="numStrokes" type="integer">
    <annotation>
      <documentation>
        Number of strokes in the flash
      </documentation>
    </annotation>
  </element>
  <element name="geometry" type="gml:PointPropertyType">
    <annotation>
      <documentation>
        The location of this lightning flash
      </documentation>
    </annotation>
  </element>
</sequence>
</extension>
</complexContent>
</complexType>
</schema></wfs:DescribeFeatureTypeResponse>

```

5.8 GetFeature Request/Response

The GetFeature Request/Response interface allows users to retrieve feature data from the WFS server for a specified feature type. The generic Client can be used for retrieving unfiltered as well as filtered data from the WFS server.

Prior to executing the following test cases load the data files in sample-requests/AIREPData into the WFS server using the following as a sample command:

```
target/bin/SoapClient http://ngen-wfsri.wx.ll.mit.edu/soap/wfs  
../sample-requests/AIREPTestData/20090911092014_001335.xml AIREP  
AIREP
```

Use similar methods to insert 7 additional files in
~/wxforge/sample-requests/AIREPTestData

5.8.1 Unfiltered Access

The file `sample-requests/AIREPQueries/mdcrUnfiltered.xml` (shown below) shows a simple OGC filter that when executed retrieves all MDCR data that is available at the server. To run the filter use the following command:

```
target/bin/SoapClient http://ngen-wfsri.wx.ll.mit.edu/soap/wfs  
../sample-requests/AIREPQueries/mdcrUnfiltered.xml CONSUMER1 CONS
```

Verify that this returns results of the type:

```
<gml:featureMember xmlns:gml="http://www.opengis.net/gml">  
  
<avwx:AircraftReport xmlns:avwx="http://www.eurocontrol.int/wxxs/1.1" xmlns:om="http://www.opengis.net/om/1.0/gml32"  
xmlns:wx="http://www.eurocontrol.int/wx/1.1" xmlns:wxont="http://wmo.int/ontologies/wx.owl#"  
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" gml:id="id0"  
xsi:schemaLocation="http://www.eurocontrol.int/wxxs/1.1 /int/eurocontrol/wxxs/1.1.0/wxxs.xsd">  
  
<avwx:aircraftReportType>AIREP</avwx:aircraftReportType>  
<avwx:aircraftId codeSpace="urn:icao:Aircraft:type">IUAX02</avwx:aircraftId>  
<avwx:airspaceWxObservation>  
  <wx:Observation gml:id="id6">  
    <om:samplingTime>  
      <gml:TimeInstant gml:id="id8">  
        <gml:timePosition>2009-08-28T03:19:00Z</gml:timePosition>  
      </gml:TimeInstant>  
    </om:samplingTime>  
    <om:procedure xlink:href="urn:fdc:icao:procedure:AircraftReport"/>  
    <om:observedProperty xlink:href="http://www.eurocontrol.int/ont/avwx/1.1/wx.owl#AirspaceWx"/>  
    <om:featureOfInterest>  
      <avwx:Airspace gml:id="id4">  
        <gml:location>  
          <gml:Point gml:id="id5" srsName="urn:ogc:def:crs:EPSG::4326">  
            <gml:pos>39.49 -78.25 6084.0</gml:pos>  
          </gml:Point>  
        </gml:location>  
      </avwx:Airspace>  
    </om:featureOfInterest>  
  </om:result>  
  <avwx:AirspaceWx gml:id="id10">  
    <avwx:windSpeed uom="kt">8</avwx:windSpeed>  
    <avwx:windDirection uom="deg">229</avwx:windDirection>  
    <avwx:airTemperature uom="C">262</avwx:airTemperature>  
    <avwx:turbulence>  
      <avwx:Turbulence gml:id="tb1"/>  
    </avwx:turbulence>  
  </avwx:AirspaceWx>  
</om:featureOfInterest>  
</om:result>  
</avwx:airspaceWxObservation>  
</om:featureMember>
```

```

    </avwx:AirspaceWx>
  </om:result>
</wx:Observation>
</avwx:airspaceWxObservation>
</avwx:AircraftReport></gml:featureMember>

```

5.8.2 Spatial Subsetting

The file `sample-requests/AIREPQueries/airepBBOX.xml` (shown below) shows a simple OGC filter that when executed retrieves all Lightning data that is available for the specified bounding box. To run the filter use the following command:

```

target/bin/SoapClient http://ngen-wfsri.wx.ll.mit.edu/soap/wfs
../sample-requests/AIREPQueries/airepBBOX.xml CONSUMER1 CONS

```

Filter Query:

```

<wfs:GetFeature service="WFS" version="1.1.0" outputFormat="text/xml; subtype=gml/3.1.1"
  xmlns:nnew="http://wx.ll.mit.edu/nnew" xmlns:wfs="http://www.opengis.net/wfs"
  xmlns:ogc="http://www.opengis.net/ogc" xmlns:gml="http://www.opengis.net/gml"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.opengis.net/wfs ../../../../../../ogc-bindings/schemas/net/opengis/wfs/1.1.0-LL/wfs.xsd">
  <wfs:Query typeName="AircraftReport">
    <ogc:Filter>
      <ogc:BBOX>
        <gml:Envelope>
          <gml:coordinates> 35.0,-88.0 40.0,-120.0 </gml:coordinates>
        </gml:Envelope>
      </ogc:BBOX>
    </ogc:Filter>
  </wfs:Query>
</wfs:GetFeature>

```

Verify that all results fall in the bounding box specified (**35.0,-88.0 40.0,-120.0**).

5.8.3 Temporal Subsetting

The file `sample-requests/AIREPQueries/airepIssueTime_2.xml` (shown below) shows a simple OGC filter that when executed retrieves all AircraftReports that falls within the specified temporal bounding box. To run the filter use the following command:

```

target/bin/SoapClient http://ngen-wfsri.wx.ll.mit.edu/soap/wfs
../sample-requests/AIREPQueries/airepIssueTime_2.xml CONSUMER1
CONS

```

Filter Query:


```

<wfs:GetFeature service="WFS" version="1.1.0" outputFormat="text/xml; subtype=gml/3.1.1"
  xmlns:nnew="http://wx.ll.mit.edu/nnew" xmlns:wfs="http://www.opengis.net/wfs"
  xmlns:ogc="http://www.opengis.net/ogc" xmlns:gml="http://www.opengis.net/gml"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.opengis.net/wfs ../..../..../ogc-bindings/schemas/net/opengis/wfs/1.1.0-LL/wfs.xsd">
  <wfs:Query typeName="AircraftReport">
    <ogc:Filter>
      <ogc:And>
        <ogc:PropertyIsGreaterThan>
          <ogc:PropertyName>wx.issueTime</ogc:PropertyName>
          <ogc:Literal>2009-09-10T12:51:34Z</ogc:Literal>
        </ogc:PropertyIsGreaterThan>
        <ogc:PropertyIsLessThanOrEqualTo>
          <ogc:PropertyName>wx.issueTime</ogc:PropertyName>
          <ogc:Literal>2009-09-17T12:51:34Z</ogc:Literal>
        </ogc:PropertyIsLessThanOrEqualTo>
      </ogc:And>
    </ogc:Filter>
  </wfs:Query>
</wfs:GetFeature>

```

Verify that results returned fall between the specified temporal bounding box.

5.9 GetFeature Subscription Using Google Earth Subscription Client

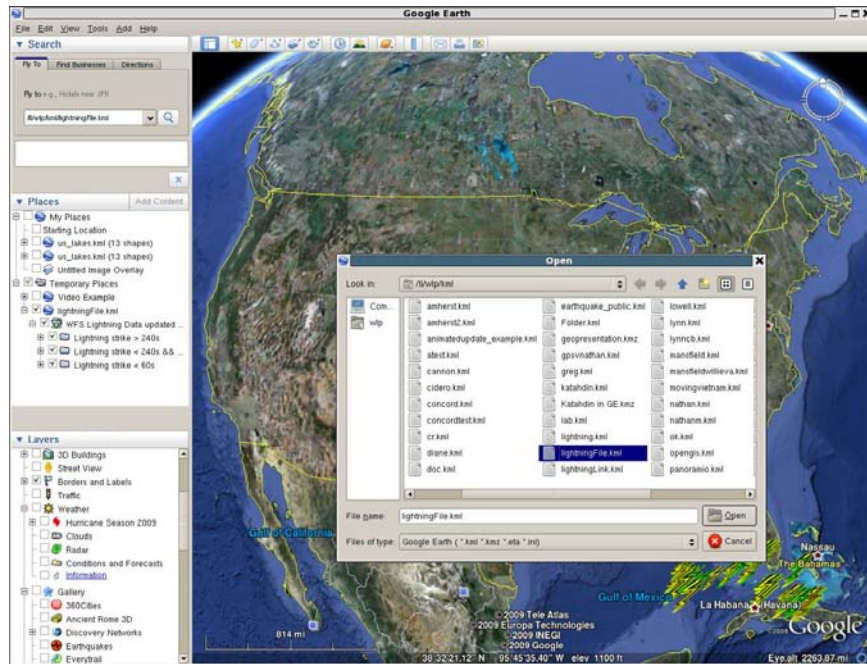
The Integrated Google Earth subscription client demonstrates the subscription capabilities of the WFSRI. The Google Earth Subscription Client establishes a subscription to the desired feature data at the WFS server, in this case Lightning data, and displays the lightning flashes.

5.9.1 Unfiltered Subscription

Go the directory where you installed Google Earth. Run Google Earth using the following command:

```
./googleearth
```

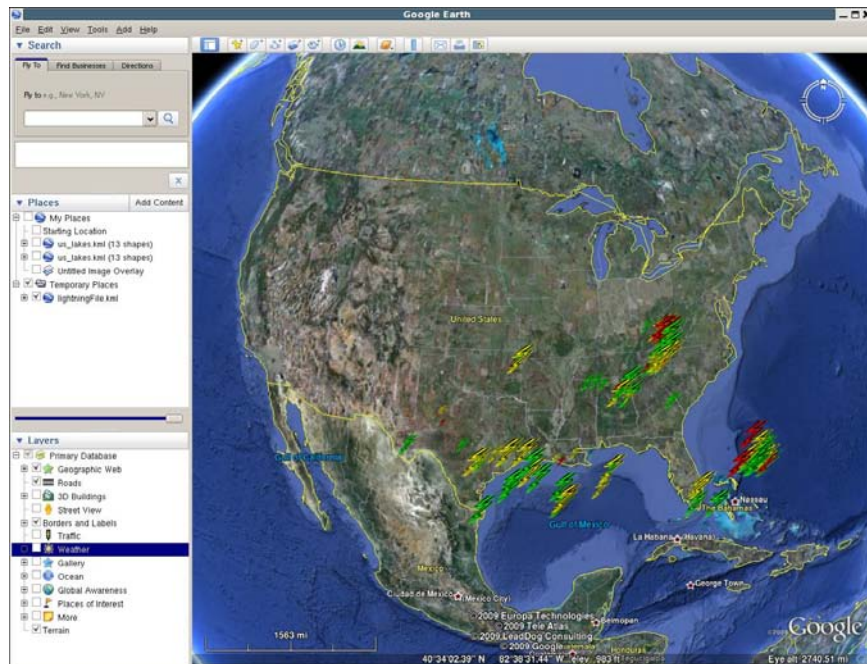
In Google Earth, load the kml file that will monitor the output of GMLToKmlTest. To do this click on **File** in Google Earth, then **Open.** Use the window that came up to browse \$HOME/kml. Click on lightningFile.kml then click the **Open** button.



In the window where GmlToKmlTest is, type

```
target/bin/GMLToKmlTest -e http://ngen-wfsri.wx.ll.mit.edu/soap/wfs -u
CONSUMER1 -p CONS
```

Lightning bolts should be visible in Google Earth soon. A sample screenshot is shown below.



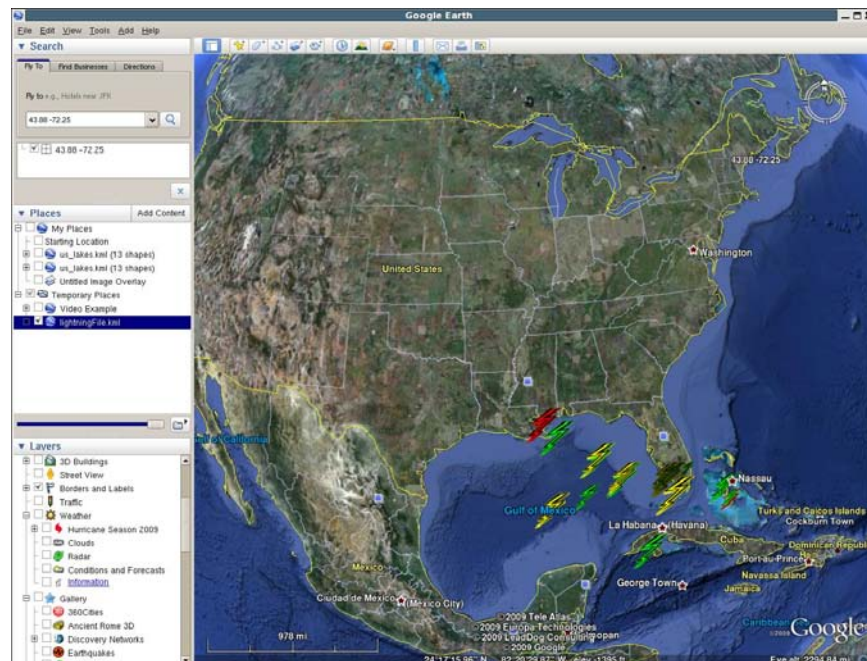
5.9.2 Spatial Filtering Subscription

To demonstrate geometric subsetting, run GMLToKmlTest with the `-r` flag. This will tell the program to look at different regions. Currently the choices are CONUS, GULF, TX, and FL.

In the window where GmlToKmlTest is, type

```
target/bin/GMLToKmlTest -e http://ngen-wfsri.wx.ll.mit.edu/soap/wfs -u
CONSUMER1 -p CONS -r FL
```

Lightning bolts over Florida should be visible. A sample screenshot is shown below.



5.10 Security

5.10.1 Verification of Unauthorized Access

Change directory to where you installed the SamplePublisherClientApp

```
(/home/nwec/wxforge/nnew/ri-client/trunk/wfsri).
```

Use the SamplePublisherClientApp script to publish to the WFS using an incorrect password.

```
target/bin/SamplePublisherClientApp -e http://ngen-
wfsri.wx.ll.mit.edu/soap/wfs -u LTNPRODUCER -p LTO
```

Here the location of the WFS is specified with the `-e` option, the producer name is specified with the `-u` option and the producer's password is specified with the `-p` option. The password for this producer should be LTN, not LTO. Verify that the server returns a “**Bad Credentials**” message.

```
ERROR: Bad credentials
```

5.10.2 Verification of Authorized Access

Use the `SamplePublisherClientApp` script with the correct password.

```
target/bin/SamplePublisherClientApp -e http://ngen-  
wfsri.wx.ll.mit.edu/soap/wfs -u LTNPRODUCER -p LTN
```

Running this script should now show a response from the service with its own valid security measures.

```
4] - Sent request [SaajSoapMessage {http://www.opengis.net/wfs}Transaction]  
2] - Received response [SaajSoapMessage xml-fragment] for request [SaajSoapMes  
Message:486] - Validating message [SaajSoapMessage xml-fragment] with actions  
eader:230] - enter processSecurityHeader()  
eader:241] - Processing WS-Security header for '' actor.  
] - Found Timestamp list element  
o:62] - Preparing to verify the timestamp  
o:66] - Current time: 2009-09-10T17:22:08.756Z  
o:69] - Timestamp created: 2009-09-10T17:22:08.741Z  
o:73] - Timestamp expires: 2009-09-10T17:27:08.741Z  
eader:343] - processHeader: total 3, prepare 1, handle 2  
Preparing to verify the timestamp  
Validation of Timestamp: Current time is 2009-09-10T17:22:08.758Z  
Validation of Timestamp: Valid creation is 2009-09-10T17:12:08.758Z  
Validation of Timestamp: Timestamp created is 2009-09-10T17:22:08.741Z  
Validation of Timestamp: Everything is ok
```

Verify that the request was sent, a response was received, and that the response was sent for the correct actor with a valid timestamp.

5.11 Client-Side Service Adaptor – CIWS Display

In the first terminal execute the script by typing:

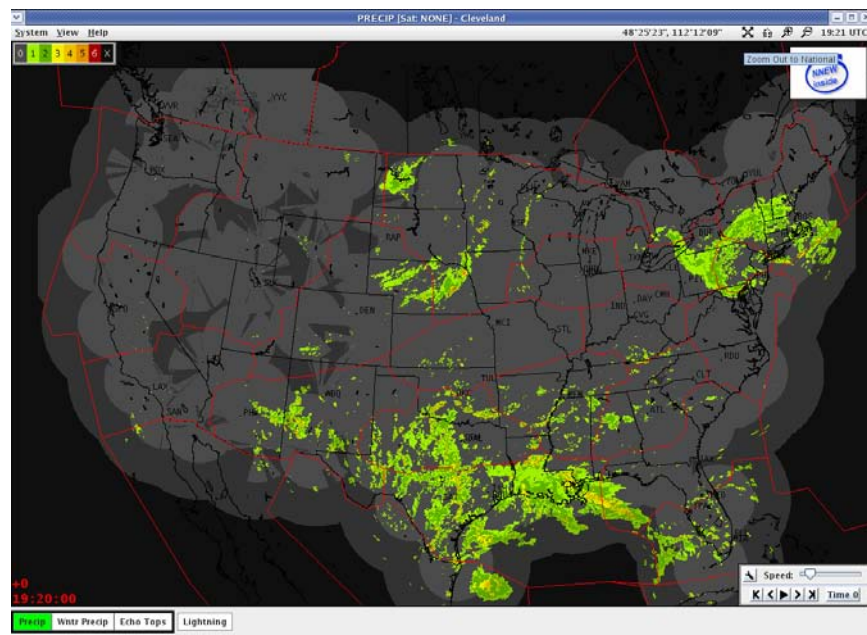
```
./run
```

In the second terminal run the “`run_LightningProcessor`” script

```
./run_LightningProcessor
```

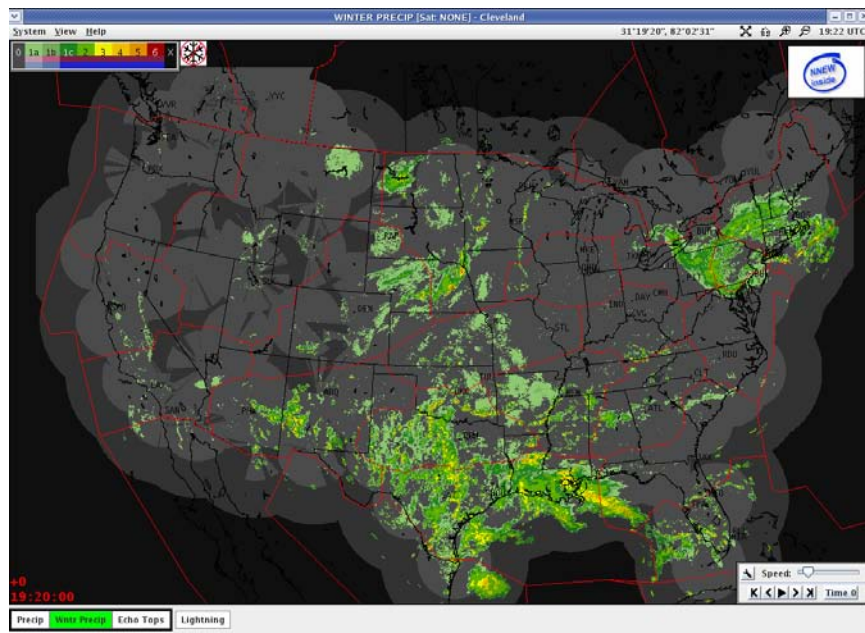
5.11.1 Precip Product

- Click on the “**Precip**” button. Click on “**Time 0**” in the Loop Controls.
- Zoom out to national view, open product dialog (right click) and uncheck show coverage limitations. Verify that the coverage limitations are shown when checked and masked out when unchecked.
- Zoom into some weather with all levels in it. Filter each level and verify that the level has been filtered.
- Close the product dialog with the close button. Verify that it has been closed.



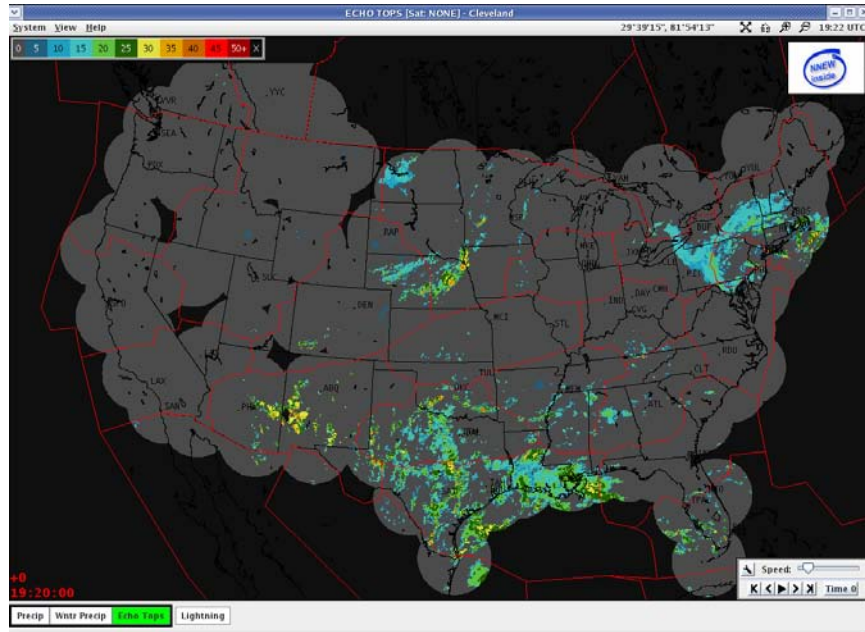
5.12 Winter Precip Product

- Click on the “**Wntr Precip**” button. Click on “**Time 0**” in the Loop Controls.
- Zoom out to national view, open product dialog (right click) and uncheck show coverage limitations. Verify that the coverage limitations are shown when checked and masked out when unchecked.
- Zoom into some weather with all levels in it. Filter each level and verify that the level has been filtered.
- Close the product dialog with the close button. Verify that it has been closed.



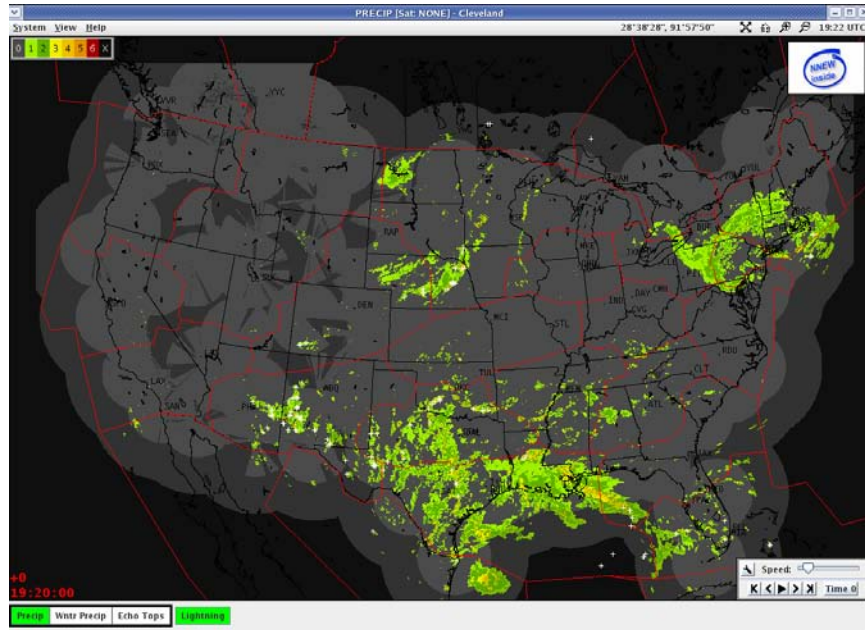
5.13 Echo Tops Product

- Click the "Echo Tops" product button. Click on "Time 0" in the Loop Controls.
- Zoom into some weather with all levels in it. Filter each level and verify that the level has been filtered.
- Close the product dialog with the close button. Verify that it has been closed.



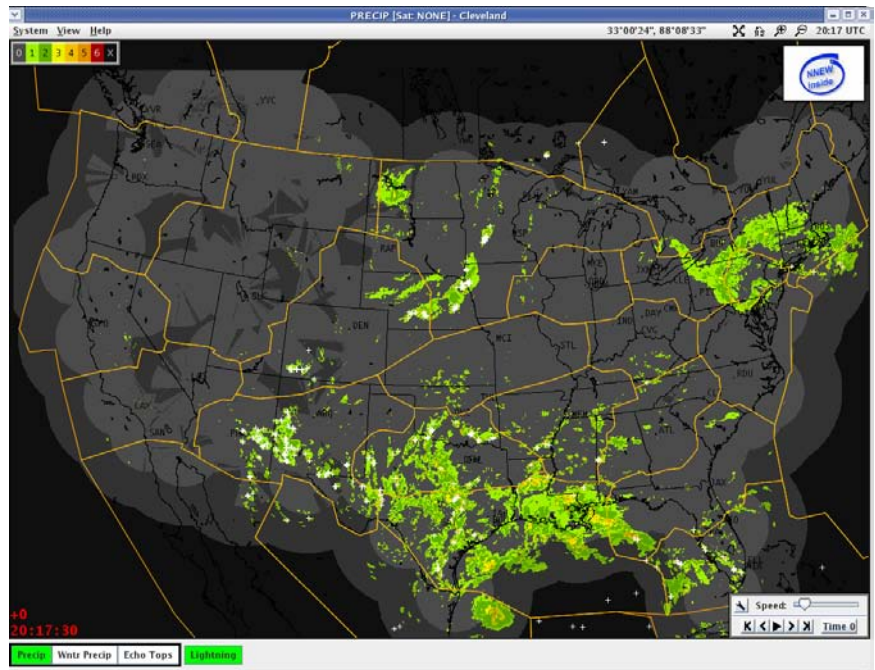
5.14 Lightning Product

- Click the "**Lightning**" product button. You may want to leave the display and lightning processor up for 30 minutes or so (to test past wx). Click on "**Time 0**" in the Loop Controls.
- Animate the loop by clicking on the "play" button in the Loop Controls. Verify that Lightning is visible in all of the past frames and time 0.
- In the product dialog check "**Time 0 only**". Verify that Lightning only appears in the time 0 frame.
- Close the product dialog with the close button. Verify that it has been closed.



5.15 AIXM Overlays

- Click on “**Time 0**” in the Loop Controls.
- Add all of the ARTCCs by clicking on the “**View**” → “**Load AIXM Overlay...**”
- In the “**Open**” dialog click on CZE.xml hold down shift and click on ZTL.xml to select all of the xml files. Click on Open.
- Verify that the ARTCCs load. The red ARTCCs will be covered by orange AIXM equivalents.
- Use the “**View**” → “**Clear AIXM Overlays**” to clear the AIXM overlays. Verify that they are cleared (ARTCCs will turn back to red)



6. NWS Data discovery and access

6.1 Introduction

As the foundation of the NWS Digital Services Program, the National Digital Forecast Database (NDFD) consists of gridded forecasts of sensible weather elements (e.g., cloud cover, maximum temperature). NDFD contains a seamless mosaic of digital forecasts from NWS field offices working in collaboration with the National Centers for Environmental Prediction (NCEP). For the purposes of this demonstration, NDFD might be considered a reasonable surrogate for the Single Authoritative Source (SAS). That said, to date NDFD weather elements have not been focused on meeting the requirements of NextGen.

The National Digital Guidance Database (NDGD), a companion to the NDFD, contains gridded weather forecasts produced by various automated techniques. Gridded forecasts of thunderstorm probability produced by the NWS' Localized Aviation MOS Program (LAMP) are one component of the NDGD.

In 2006 the NWS declared operational a web service that uses Simple Object Access Protocol (SOAP) to deliver NDFD and NDGD data to customers and partners. This web service does not follow OGC standards. It predates many of the standards that are emerging for NextGen. It is largely focused on delivering data for a single point, not a grid or a flight path. That said, the NDFD SOAP service is operational, and the forecasts and guidance it delivers are official. The NWS web farm routinely services millions of hits per day for the NDFD SOAP service. This service can be registered in the WellGEO registry/repository software.

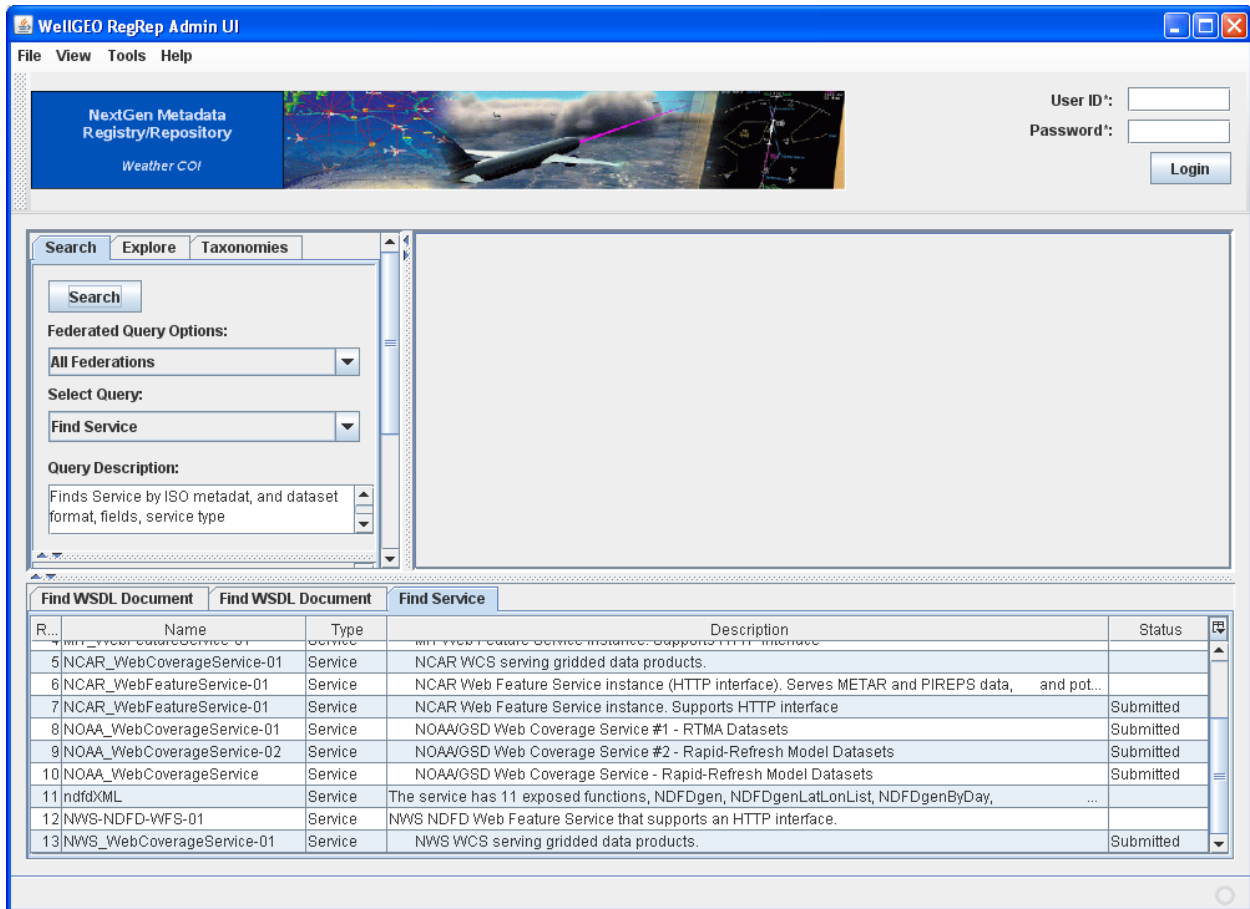
6.2 Test Environment and Setup

Setup for this test is described in Section 2.2, Demonstration Applications.

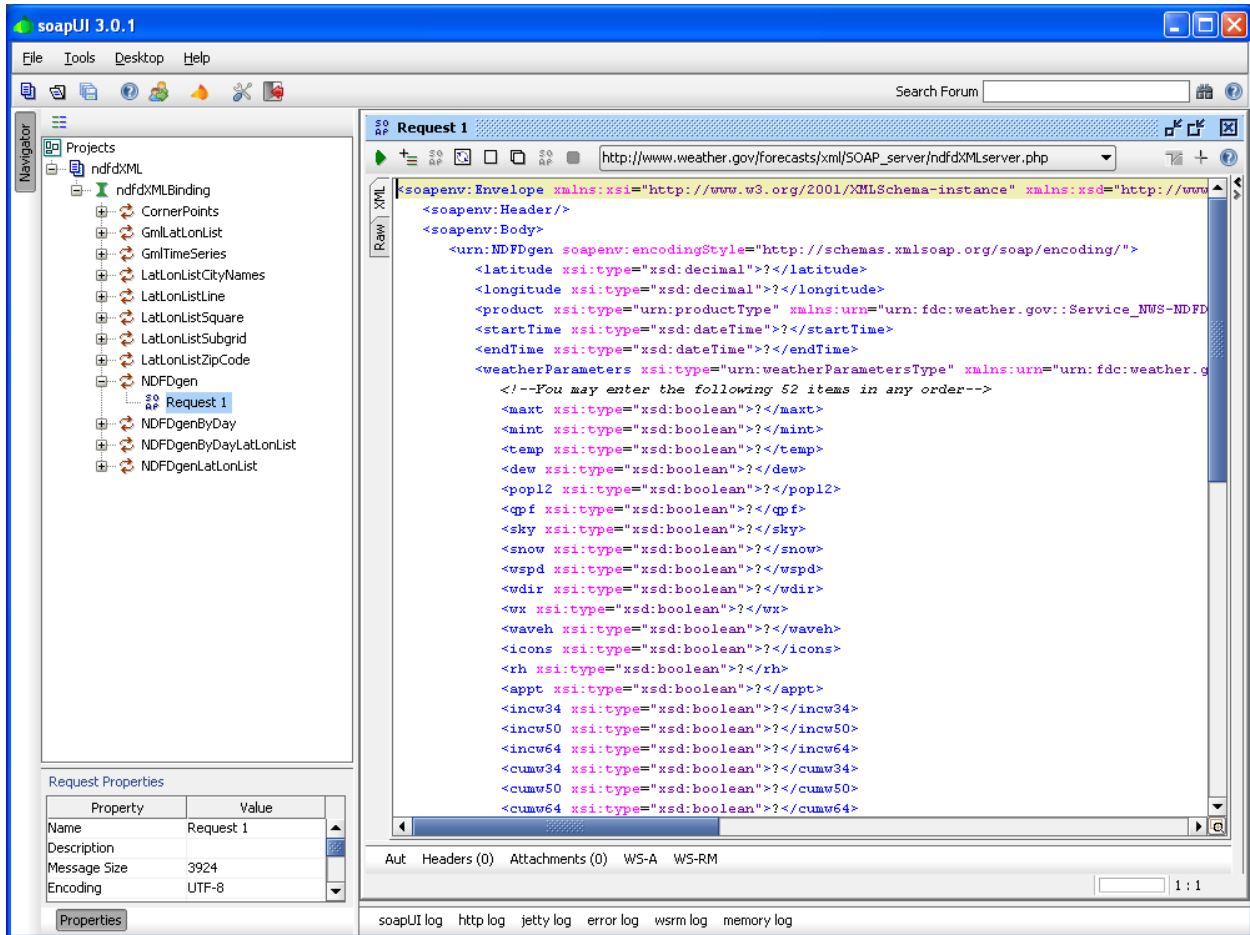
6.3 Discovery of NDFD/NDGD SOAP Service

This test uses WellGEO to find the operational NWS NDFD SOAP service. Then, it uses soapUI to show the capabilities of the NDFD SOAP service.

1. Launch the Registry Admin UI **version 4.4** as described in section 3.1.4, above.
2. Click on the “**Search**” tab in upper left corner of the UI to access the Search Tool panel.
3. Select “**All Federations**” in the “**Federated Query Options**” dropdown menu.
4. Select “**Find Service**” in the “**Select Query**” combo box.
5. Click the “**Search**” button at the top of the Search Tool panel. An unfiltered list of all the registered services will appear. Click on the “**Name**” column heading to sort the datasets by name / Name (alphabetical order).
6. Verify that the list of services returned includes “**ndfdXML.**”



7. Right click the entry for “**ndfdXML**” and select “**Download.**”
Note: This may not work due to missing file. See next instruction:
8. Find the endpoint in the Regrep and paste the endpoint into the browser. Click on the “**WSDL**” link. Copy the wsdl URL and save it to paste it into the soapUI.
9. Launch soapUI, and select “**File**” → “**New soapUI Project.**” Copy the wsdl URL into the Initial WSDL box.
10. Verify that the list of operations includes “**NDFDgen**” and “**NDFDgenByDay.**” This list shows the 12 SOAP operations that customers and partners can request from the service. NDFDgen and NDFDgenByDay are the two primary operations for the NWS' operational NDFD SOAP service. NDFDgen will be used later to query data.
11. Expand the operation named “**NDFDgen**”. Double click “**Request 1**”. Verify the list of weatherParameters includes “**wspd**” and “**wdir.**” SoapUI allows one to see the calling parameters needed for a query (lat/lon, time range, weather element). It also shows the list of various NDFD and NDGD weather elements that are available.



6.4 SOAP Query of NDFD/NDGD Data

To support this demo, the NWS has set up a simple browser interface. It can query NDFD data for the closest gridpoint from the operational NWS servers. Output is displayed in either XML or tabular form.

1. Open the following URL in a web browser:

http://www.weather.gov/forecasts/xml/SOAP_server/NextGenDemo.htm

2. Select “**Digital Weather Markup Language (DWML)**”
3. Enter the latitude and longitude for one of the following “windy” Operational Evolution Partnership (OEP) airports. Coordinates listed here are coarse, but sufficient to capture the correct gridpoint.

Identifier	Latitude	Longitude
KSFO	37.62	-122.37
KBOS	42.36	-71.02

KLGA	40.78	-73.88
KDFW	32.90	-97.02

4. If desired, change “**Start Time**” and “**End Time.**” The preset values will query all forecast periods available in NDFD.
5. Select “**Wind Speed**” and “**Wind Direction.**”

6. Click the “**Submit**” button at the bottom of the page. XML data will be returned. Valid periods, a time series of forecast wind speeds, and a time series of forecast wind directions will be recognizable.

The screenshot shows a web browser window displaying the National Weather Service's Weather Information Database Access Page. The browser's address bar shows the URL: `http://www.weather.gov/forecasts/xml/SOAP_server/NextGenDemo.htm`. The page header includes the NOAA logo and the text "National Oceanic and Atmospheric Administration's National Weather Service". A navigation bar contains links for "Site Map", "News", "Organization", "Search", "NWS", and "All NOAA".

The main content area is titled "National Weather Service Weather Information Database Access Page". It features a "Data Encoding Format" section with two radio buttons: "Digital Weather Markup Language (DWML)" (selected) and "Text Table". Below this is a "Location (KAID = 38.9417 -77.4575)" section with input fields for "Latitude" (42.36) and "Longitude" (-71.02). The "Times" section has input fields for "Start Time" (2004-01-01T00:00:00) and "End Time" (2009-09-10T00:00:00). The "Variable" section contains a grid of checkboxes for various weather variables, with "Wind Speed" and "Wind Direction" checked.

A "Submit" button is located at the bottom left of the form area. On the left side of the page, there is a vertical navigation menu with categories such as "Local forecast by 'City, St'", "XML RSS Feeds", "Warnings", "Observations", "Forecasts", "Text Messages", "Forecast Models", "Climate", "Weather Safety", and "Education/Outreach Information Center".

```

<dwml version="1.0" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://www.nws.noaa.gov/forecasts/xml/DWMLgen/schema/DWML.xsd"
>
  <head>
    <product srsName="WGS 1984" concise-name="time-series" operational-mode="official">
      <title>NOAA's National Weather Service Forecast Data</title>
      <field>meteorological</field>
      <category>forecast</category>
      <creation-date refresh-frequency="PT1H">2009-09-09T17:51:05Z</creation-date>
    </product>
    <source>
      <more-information>http://www.nws.noaa.gov/forecasts/xml</more-information>
      <production-center>Meteorological Development Laboratory<sub-center>Product Generation
Branch</sub-center></production-center>
      <disclaimer>http://www.nws.noaa.gov/disclaimer.html</disclaimer>
      <credit>http://www.weather.gov/</credit>
      <credit-logo>http://www.weather.gov/images/xml_logo.gif</credit-logo>
      <feedback>http://www.weather.gov/feedback.php</feedback>
    </source>
  </head>
  <data>
    <location>
      <location-key>point1</location-key>
      <point latitude="42.36" longitude="-71.02"/>
    </location>
    <moreWeatherInformation applicable-
location="point1">http://forecast.weather.gov/MapClick.php?textField1=42.36&textfield2=-
71.02</moreWeatherInformation>
    <time-layout time-coordinate="local" summarization="none">
      <layout-key>k-p3h-n5-1</layout-key>
      <start-valid-time>2009-09-09T14:00:00-04:00</start-valid-time>
      <start-valid-time>2009-09-09T17:00:00-04:00</start-valid-time>
      <start-valid-time>2009-09-09T20:00:00-04:00</start-valid-time>
      <start-valid-time>2009-09-09T23:00:00-04:00</start-valid-time>
      <start-valid-time>2009-09-10T02:00:00-04:00</start-valid-time>
    </time-layout>
    <parameters applicable-location="point1">
      <wind-speed type="sustained" units="knots" time-layout="k-p3h-n5-1">
        <name>Wind Speed</name>
        <value>15</value>
        <value>14</value>
        <value>10</value>
        <value>8</value>
        <value>11</value>
      </wind-speed>
      <direction type="wind" units="degrees true" time-layout="k-p3h-n5-1">
        <name>Wind Direction</name>
        <value>50</value>
        <value>60</value>
        <value>60</value>
        <value>60</value>
        <value>50</value>
      </direction>
    </parameters>
  </data>
</dwml>

```

7. Use the browser's **“Back”** button to return to the menu page.
8. Select **“Text Table”**
9. Enter the latitude and longitude for one of the following “thunderstorm-prone” OEP airports.

Identifier	Latitude	Longitude
KFLL	26.07	-80.15
KATL	33.63	-84.45
KORD	41.98	-87.93

KDFW	32.90	-97.02
------	-------	--------

10. Select “LAMP Thunderstorm Probabilities.” only

The screenshot shows the National Weather Service Weather Information Database Access Page. The page is titled "National Weather Service Weather Information Database Access Page". It features a search form with the following fields and options:

- Data Encoding Format:** Radio buttons for "Digital Weather Markup Language (DWML)" and "Text Table". "Text Table" is selected.
- Location (KAID = 38.9417 -77.4575):** Text input fields for "Latitude" (26.07) and "Longitude" (-80.15).
- Times:** Text input fields for "Start Time" (2004-01-01T00:00:00) and "End Time" (2009-09-10T00:00:00).
- Variable:** A list of checkboxes for various weather variables. "LAMP Thunderstorm Probabilities" is checked. Other variables include Wind Speed, Wind Direction, LAMP Categorical Thunderstorm, MaxT, MinT, Hourly Temperatures, Dewpoint Temperature, QPF, PoP12, Sky Cover, Weather, RTMA Precipitation, RTMA GOES Cloud Amount, RTMA Dewpoint Temperature, RTMA Temperature, RTMA Wind Direction, RTMA Wind Speed, Hazards, and Wind Gust.

A "Submit" button is located at the bottom of the form.

11. Click the “Submit” button at the bottom of the page. A table of valid times and thunderstorm probabilities will be returned.


```

latitude = 26.07 longitude = -80.15

element, unit, refTime, validTime, (26.070000, -80.150000)
TSTM02, [%], 200909091700, 200909092000, 49.000
TSTM02, [%], 200909091700, 200909092100, 48.000
TSTM02, [%], 200909091700, 200909092200, 43.000
TSTM02, [%], 200909091700, 200909092300, 32.000
TSTM02, [%], 200909091700, 200909100000, 22.000
TSTM02, [%], 200909091700, 200909100200, 12.000
TSTM02, [%], 200909091700, 200909100400, 7.000
TSTM02, [%], 200909091700, 200909100600, 5.000
TSTM02, [%], 200909091700, 200909100800, 5.000
TSTM02, [%], 200909091700, 200909101000, 7.000
TSTM02, [%], 200909091700, 200909101200, 9.000
TSTM02, [%], 200909091700, 200909101400, 13.000
TSTM02, [%], 200909091700, 200909101600, 20.000
TSTM02, [%], 200909091700, 200909101800, 32.000

```

6.5 Web Coverage Service Query of NDFD/NDGD Data

To support this demo, the NWS has set up a Web Coverage Service (WCS) based on the reference implementation generated at NCAR. Our exposed endpoint is <http://140.90.90.88/wcs>. The NWS has created four test Coverages (Convective Probability, Wind Direction, Wind Speed and Wind Gust) for the purposes of this demonstration. The following guidelines have been created to access and use our WCS. These guidelines assume the use of soapUI.

1. Launch soapUI, and select “File” → “New soapUI Project.” A “New soapUI Project” GUI will appear.
2. Next to “Project Name,” type:
NWS WCS
3. Next to “Initial WSDL/WADL,” type:
<http://140.90.90.88/wcs?wsdl>
Check to make sure that only the “Create Requests” box has a check next to it. If not, please make sure this is the only box with a check in it. Then hit the “OK” button. This will create a project called “NWS WSC”.
4. Three features will be offered as apart of the WCS. These include describeCoverage, getCapabilities and getCoverage.
5. Click on the “+” symbol for each of these features, and you will reveal a “SOAP Request 1”. These soap requests serve as the interface between the user, and our WCS (where the data resides).
6. Double click on the “SOAP Request 1” associated with **getCapabilitiesOperation**. This will open a 2-panel GUI interface. The left-hand side of the interface contains the soap request being sent to our WCS. The right-hand side of the interface contains the soap response from our WCS server. There are no fields that need to be populated with this request. This is a simple request to find out what data is available and more background information about our WCS. Before hitting the green submit arrow, please take a look at the URL setting in the top center part of the 2-pane GUI. If this is not set to “<http://140.900.90.88/wcs>,” please edit the URL and set it to “<http://140.90.90.88/wcs>.” Then click on the green arrow (Submit request to a specified endpoint URL). After execution, a getCapabilities soap response will appear in the right-hand pane indicating some background information about who’s offering the service, and what data is available via our WCS. The most

important information that is contained in this soap response is the "**ns2:parameter name="Identifier"**" parameter. This will provide ns2:Values parameter indicating what type of data is available. The "**urn**" value from this soap response is an important input to the second request.

7. Double click on the "**SOAP Request 1**" associated with the **describeCoverageOption**. This will open a 2-panel GUI interface. The left-hand side of the interface contains the describeCoverage soap request being sent to our WCS. The right-hand side of the interface contains the soap response from our WCS server. There is one input field ("**ns:Identifier**") that needs to be entered before the soap request can be submitted.
 - a. To see the "**Convective Probability**" Coverage:
 - Replace the "?" by the "**ns:Identifier**" tag with "`urn:fdc:weather.gov:Dataset:ConvectiveProbability-LAMP`". Once this is in place, click on the green arrow (Submit request to a specified endpoint URL).
 - b. To see the "**Wind Direction**" Coverage:
 - Replace the "?" by the "**ns:Identifier**" tag with "`urn:fdc:weather.gov:Dataset:WindDirection-NDFD-CONUS-1`". Once this is in place, click on the green arrow (Submit request to a specified endpoint URL).
 - c. To see the "**Wind Speed**" Coverage:
 - Replace the "?" by the "**ns:Identifier**" tag with "`urn:fdc:weather.gov:Dataset:WindSpeed-NDFD-CONUS-1`". Once this is in place, click on the green arrow (Submit request to a specified endpoint URL).
 - d. To see the "**Wind Gust**" Coverage:
 - Replace the "?" by the "**ns:Identifier**" tag with "`urn:fdc:weather.gov:Dataset:WindGust-NDFD-CONUS-1`". Once this is in place, click on the green arrow (Submit request to a specified endpoint URL).

After execution, a describeCoverage soap response will appear in the right-hand pane showing information about the title of the data, some applicable keywords, the identifier of the data, the spatial domain (lat/lon boundaries) for the data, what time(s) are available for the data, and what supported output format is available. The following fields would normally need to be filled in before a getCoverage soap request can be made (See list below). However, for the purposes of this demonstration, the actual soap requests will be provided for each Coverage (See 8 below).

1. ns1:Identifier
2. crs
(Please use urn:ogc:def:crs:OGC:2:84)
3. dimensions

- (Please use 2)
- 4. ns1:LowerCorner for Lat/Long
(See available lat/long range specified in describeCoverage soap response)
- 5. ns1:UpperCorner for Lat/Long
- 6. gml:timeposition
(Please comment this line out. For example: <!--gml:timePosition frame="#ISO-8601" calendarEraName="" indeterminatePosition=""?>?</gml:timePosition-->)
- 7. ns:BeginPosition for start time
(See available options from the describeCoverage soap response)
- 8. ns:EndPosition for the end time
(See available options from the describeCoverage soap response)
- 9. FieldSubset ns1:Identifier
(See available option from the describeCoverage soap response)
- 10. ns:Output for the output format
(Please use application/netcdf4).

1. Double click on the "**SOAP Request 1**" associated with the **getCoverageOption**. This will open a 2-panel GUI interface. The left-hand side of the interface contains the getCoverage soap request being sent to our WCS. The right-hand side of the interface contains the soap response from our WCS server. The fields highlighted in bold from above must be populated in the soap request (left-side pane) and change the time stamps for the request to be a date copy/pasted from the results of "**describeCoverage**" before submitting the getCoverage soap request. For the purposes of this demonstration, re-populated soap requests have been generated for each Coverage type. Copy and paste this information into the getCoverage soap request (left-side window pane). Then click the on the green arrow (Submit request to a specified endpoint URL). An attachment should show up in the soap response (right-side pane).
2. To export the attachment, click once on the word "**Attachments**" at the bottom of the soap response window pane. A panel with "**Name**", "**Content type**", etc will appear. Below that line is the netCDF4 file. Click once anywhere on this line. A green tab will appear that allows for the export of the selected attachment to a file. Click that tab once, and an "**Export Attachment**" GUI will appear. Save the file to a desired location. A number of programs can be used to view the netCDF4 file, including nview and ncdump.

- a. To see the "**Convective Probability**" Coverage:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:ns="http://www.opengis.net/wcs/1.1"
xmlns:ns1="http://www.opengis.net/ows/1.1"
xmlns:gml="http://www.opengis.net/gml">
  <soap:Header/>
  <soap:Body>
    <ns:GetCoverage service="WCS" version="1.1.2">
      <ns1:Identifier>urn:fdc:weather.gov:Dataset:ConvectiveProbability-LAMP</ns1:Identifier>
      <ns:DomainSubset>
        <ns1:BoundingBox crs="urn:ogc:def:crs:OGC:2:84"
dimensions="2">
          <ns1:LowerCorner>-125.0 20.6</ns1:LowerCorner>
          <ns1:UpperCorner>-60.0 49.0</ns1:UpperCorner>
        </ns1:BoundingBox>
      </ns:DomainSubset>
    </ns:GetCoverage>
  </soap:Body>
</soap:Envelope>
```

```

        <!-- Optional: -->
        <ns: TemporalSubset>
            <!-- You have a CHOICE of the next 2 items at this level-->
            <!-- gml:timePosition frame="#ISO-8601" calendarEraName="?"
indeterminatePosition="?"></gml:timePosition-->
            <ns: TimePeriod>
                <ns: BeginPosition>2009-09-
03T17:00:00.000Z</ns: BeginPosition>
                <ns: EndPosition>2009-09-
03T17:00:00.000Z</ns: EndPosition>
            <!-- Optional: -->
            <!-- <ns: TimeResolution>?</ns: TimeResolution-->
            </ns: TimePeriod>
        </ns: TemporalSubset>
    </ns: DomainSubset>
    <!-- Optional: -->
    <ns: RangeSubset>
        <!-- 1 or more repetitions: -->
        <ns: FieldSubset>
            <ns1: Identifier>Thunderstorm_probability</ns1: Identifier>
            <!-- Optional: -->
            <ns: InterpolationType>?</ns: InterpolationType>
            <!-- Zero or more repetitions: -->
            <ns: AxisSubset>
                <ns: Identifier>?</ns: Identifier>
                <!-- 1 or more repetitions: -->
                <ns: Key>?</ns: Key>
            </ns: AxisSubset>
        </ns: FieldSubset>
    </ns: RangeSubset>
    <ns: Output format="application/netcdf4" store="false">
        <!-- Optional: -->
        <ns: GridCRS gml:id="?">
            <!-- Optional: -->
            <gml:srsName codeSpace="?"></gml:srsName>
            <ns: GridBaseCRS>?</ns: GridBaseCRS>
            <!-- Optional: -->
        </ns: GridType>urn:ogc:def:method:WCS:1.1:2dSimpleGrid</ns: GridType>
        <!-- Optional: -->
        <ns: GridOrigin>0 0</ns: GridOrigin>
        <ns: GridOffsets>?</ns: GridOffsets>
        <!-- Optional: -->
    </ns: GridCS>urn:ogc:def:cs:OGC:0.0:Grid2dSquareCS</ns: GridCS>
    </ns: GridCRS>
    </ns: Output>
    </ns: GetCoverage>
</soap: Body>
</soap: Envelope>

```

- b. To see the “*Wind Direction*” Coverage:

```

<soap: Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:ns="http://www.opengis.net/wcs/1.1"
xmlns:ns1="http://www.opengis.net/ows/1.1"
xmlns:gml="http://www.opengis.net/gml">
  <soap: Header/>
  <soap: Body>
    <ns: GetCoverage service="WCS" version="1.1.2">
      <ns1: Identifier>urn:fdc:weather.gov:Dataset:WindDirection-NDFD-
CONUS-1</ns1: Identifier>
      <ns: DomainSubset>
        <ns1: BoundingBox crs="urn:ogc:def:crs:OGC:2:84"
dimensions="2">
          <ns1: LowerCorner>-125.0 20.6</ns1: LowerCorner>
          <ns1: UpperCorner>-60.0 49.0</ns1: UpperCorner>
        </ns1: BoundingBox>
        <!-- Optional: -->
      <ns: TemporalSubset>
        <!-- You have a CHOICE of the next 2 items at this level-->

```

```

        <!-- gml:timePosition frame="#ISO-8601" calendarEraName="?"
indeterminatePosition="?"?></gml:timePosition-->
        <ns:TimePeriod>
          <ns:BeginPosition>2009-08-
22T21:00:00.000Z</ns:BeginPosition>
          <ns:EndPosition>2009-08-
22T21:00:00.000Z</ns:EndPosition>
          <!-- Optional:-->
          <!--<ns:TimeResolution?></ns:TimeResolution-->
        </ns:TimePeriod>
      </ns:TemporalSubset>
    </ns:DomainSubset>
    <!-- Optional:-->
    <ns:RangeSubset>
      <!-- 1 or more repetitions:-->
      <ns:FieldSubset>

<ns1:Identifier>Wind_recti on_fro m_w h_i ch_b l o w_i n g</ns1:Identifier>
      <!-- Optional:-->
      <ns:InterpolationType?></ns:InterpolationType>
      <!-- Zero or more repetitions:-->
      <ns:AxisSubset>
        <ns:Identifier?></ns:Identifier>
        <!-- 1 or more repetitions:-->
        <ns:Key?></ns:Key>
      </ns:AxisSubset>
    </ns:FieldSubset>
  </ns:RangeSubset>
  <ns:Output format="applicati on/netcdf4" store="false">
    <!-- Optional:-->
    <ns:GridCRS gml:id="?">
      <!-- Optional:-->
      <gml:srsName codeSpace="?"></gml:srsName>
      <ns:GridBaseCRS?></ns:GridBaseCRS>
      <!-- Optional:-->

<ns:GridType>urn:ogc:def:method:WCS:1.1:2dSimpleGrid</ns:GridType>
      <!-- Optional:-->
      <ns:GridOrigin>0 0</ns:GridOrigin>
      <ns:GridOffsets?></ns:GridOffsets>
      <!-- Optional:-->

<ns:GridCS>urn:ogc:def:cs:OGC:0.0:Grid2dSquareCS</ns:GridCS>
      </ns:GridCRS>
    </ns:Output>
  </ns:GetCoverage>
</soap:Body>
</soap:Envelope>

```

- c. To see the “*Wind Speed*” Coverage:

```

<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:ns="http://www.opengis.net/wcs/1.1"
xmlns:ns1="http://www.opengis.net/ows/1.1"
xmlns:gml="http://www.opengis.net/gml">
  <soap:Header/>
  <soap:Body>
    <ns:GetCoverage service="WCS" version="1.1.2">
      <ns1:Identifier>urn:fdc:weather.gov:Dataset:WindSpeed-NDFD-
CONUS-1</ns1:Identifier>
      <ns:DomainSubset>
        <ns1:BoundingBox crs="urn:ogc:def:crs:OGC:2:84"
dimensions="2">
          <ns1:LowerCorner>-125.0 20.6</ns1:LowerCorner>
          <ns1:UpperCorner>-60.0 49.0</ns1:UpperCorner>
        </ns1:BoundingBox>
        <!-- Optional:-->
      <ns:TemporalSubset>
        <!-- You have a CHOICE of the next 2 items at this level-->

```

```

        <!-- gml:timePosition frame="#ISO-8601" calendarEraName="?"
indeterminatePosition="?"?>/gml:timePosition-->
        <ns:TimePeriod>
            <ns:BeginPosition>2009-08-
22T21:00:00.000Z</ns:BeginPosition>
            <ns:EndPosition>2009-08-
22T21:00:00.000Z</ns:EndPosition>
            <!-- Optional: -->
            <!-- <ns:TimeResolution>?</ns:TimeResolution-->
        </ns:TimePeriod>
    </ns:TemporalSubset>
</ns:DomainSubset>
<!-- Optional: -->
<ns:RangeSubset>
    <!-- 1 or more repetitions: -->
    <ns:FieldSubset>
        <ns1:Identifier>Wind_speed</ns1:Identifier>
        <!-- Optional: -->
        <ns:InterpolationType>?</ns:InterpolationType>
        <!-- Zero or more repetitions: -->
        <ns:AxisSubset>
            <ns:Identifier>?</ns:Identifier>
            <!-- 1 or more repetitions: -->
            <ns:Key>?</ns:Key>
        </ns:AxisSubset>
    </ns:FieldSubset>
</ns:RangeSubset>
<ns:OutputFormat="application/netcdf4" store="false">
    <!-- Optional: -->
    <ns:GridCRS gml:id="?">
        <!-- Optional: -->
        <gml:srsName codeSpace="?">?</gml:srsName>
        <ns:GridBaseCRS>?</ns:GridBaseCRS>
        <!-- Optional: -->
    </ns:GridType>urn:ogc:def:method:WCS:1.1:2dSimpleGrid</ns:GridType>
    <!-- Optional: -->
    <ns:GridOrigin>0 0</ns:GridOrigin>
    <ns:GridOffsets>?</ns:GridOffsets>
    <!-- Optional: -->
</ns:GridCS>urn:ogc:def:cs:OGC:0.0:Grid2dSquareCS</ns:GridCS>
    </ns:GridCRS>
</ns:Output>
</ns:GetCoverage>
</soap:Body>
</soap:Envelope>

```

- d. To see the “*Wind Gust*” Coverage:

```

<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:ns="http://www.opengis.net/wcs/1.1"
xmlns:ns1="http://www.opengis.net/ows/1.1"
xmlns:gml="http://www.opengis.net/gml">
    <soap:Header/>
    <soap:Body>
        <ns:GetCoverage service="WCS" version="1.1.2">
            <ns1:Identifier>urn:fdc:weather.gov:Dataset:WindGust-NDFD-CONUS-
1</ns1:Identifier>
            <ns:DomainSubset>
                <ns1:BoundingBox crs="urn:ogc:def:crs:OGC:2:84"
dimensions="2">
                    <ns1:LowerCorner>-125.0 20.6</ns1:LowerCorner>
                    <ns1:UpperCorner>-60.0 49.0</ns1:UpperCorner>
                </ns1:BoundingBox>
                <!-- Optional: -->
            <ns:TemporalSubset>
                <!-- You have a CHOICE of the next 2 items at this level-->
                <!-- gml:timePosition frame="#ISO-8601" calendarEraName="?"
indeterminatePosition="?"?>/gml:timePosition-->

```

```

        <ns: TimePeriod>
          <ns: BeginPosition>2009-09-
03T18:00:00.000Z</ns: BeginPosition>
          <ns: EndPosition>2009-09-
03T18:00:00.000Z</ns: EndPosition>
          <!-- Optional: -->
          <!--<ns: TimeResolution>?</ns: TimeResolution-->
        </ns: TimePeriod>
      </ns: TemporalSubset>
    </ns: DomainSubset>
    <!-- Optional: -->
    <ns: RangeSubset>
      <!-- 1 or more repetitions: -->
      <ns: FieldSubset>
        <ns1: Identifier>Wind_speed_gust</ns1: Identifier>
        <!-- Optional: -->
        <ns: InterpolationType>?</ns: InterpolationType>
        <!-- Zero or more repetitions: -->
        <ns: AxisSubset>
          <ns: Identifier>?</ns: Identifier>
          <!-- 1 or more repetitions: -->
          <ns: Key>?</ns: Key>
        </ns: AxisSubset>
      </ns: FieldSubset>
    </ns: RangeSubset>
    <ns: OutputFormat="application/netcdf4" store="false">
      <!-- Optional: -->
      <ns: GridCRS gml:id="?">
        <!-- Optional: -->
        <gml:srsName codeSpace="?">?</gml:srsName>
        <ns: GridBaseCRS>?</ns: GridBaseCRS>
        <!-- Optional: -->
      </ns: GridCRS>
      <ns: GridType>urn:ogc:def:method:WCS:1.1:2dSimpleGrid</ns: GridType>
      <!-- Optional: -->
      <ns: GridOrigin>0 0</ns: GridOrigin>
      <ns: GridOffsets>?</ns: GridOffsets>
      <!-- Optional: -->
    </ns: GridCRS>
  </ns: Output>
</ns: GetCoverage>
</soap: Body>
</soap: Envelope>

```

3. Please cut and paste this information into the soap request by referring to the information returned in the describeCoverage soap response. Once all the fields described above have been entered (as well as valid datetimes), then click on the green arrow (Submit request to a specified endpoint URL). After execution, a getCoverage soap response will appear in the right-hand pane showing whether or not there was a successful request. If successful, there should be an attachment containing a netCDF4 file. Look for the word "**Attachments**" towards to lower left-hand corner of the getCoverage soap response (right-side pane). Click once on the word "**Attachments**". A panel with "**Name**", "**Content type**", etc will appear. Below that line is the netCDF4 file. Click once anywhere on this line. A green tab will appear that allows for the export of the selected attachment to a file. Click that tab once, and an "**Export Attachment**" GUI will appear. Save the file to a desired location. A number of programs can be used to view the netCDF4 file, including ncview and ncdump. (toolUI was used during the dry runs)

7. Implementation Verification – Flight Hazard Service and High Level Service Capability

A web service composed by the orchestration of several atomic web services has been developed to demonstrate the retrieval of vertical and horizontal views of hazards along a trajectory. This test demonstrates 4D trajectory/corridor capabilities and a higher-level service capability for coverage/gridded 2D/3D data. The addition of non-gridded/feature data is planned for future releases. The main goals of this development are:

1. Demonstrate trajectory-based retrieval
 - Validation of corridor specification
 - Retrieval of multiple variables of interest
 - Baseline quantification of system performance
2. Demonstrate the composition of fine-grained services
 - Filter data within the cube
 - Allow re-use by other clients
 - Support further composition
 - Serve as an initial model for DSS services

The test cases in this chapter cover these run-time usage scenarios.

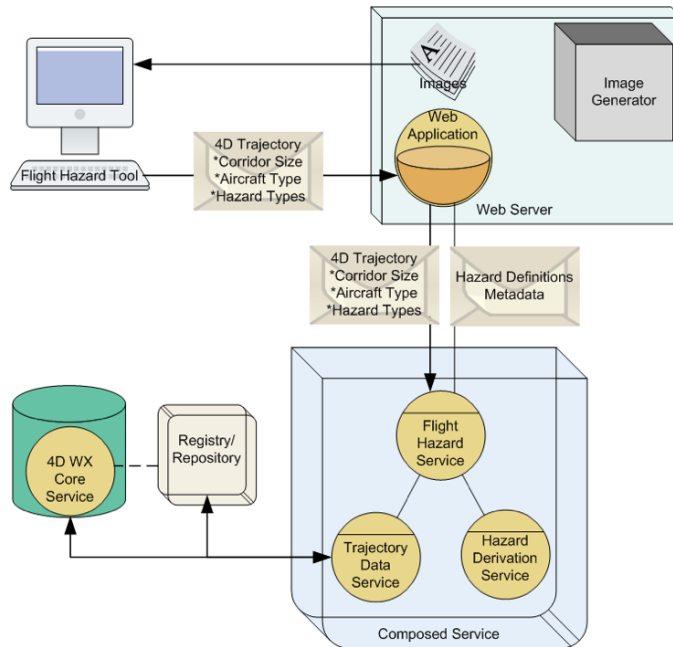
7.1 Test Environment and Setup

7.1.1 Client Software

The tests defined in this document use the following client software:

- **A Web Browser UI**

The Flight Hazard Tool UI is intended to demonstrate how a trajectory-based planning tool might get its data from the 4D Wx Data Cube. The tool runs in a web browser. Rather than retrieve entire raw datasets to display, it retrieves information about hazards along a proposed flight trajectory. The output information is a dynamically generated depiction of hazard boundaries in space and time by a server-side process, which rely on the data provided by a higher-level Flight Hazard Service.



- **SoapUI 2.5.1**

The high-level Flight Hazard web service and the services part of the orchestration/composition are exposed and can be accessed by any client using the SOAP interface. We will run several soap requests to test service accessibility and atomicity.

- **ToolsUI 4.0**

We will use the Netcdf tool to inspect some of the derived data previously created from running the tests on the SoapUI.

7.1.2 Composite Services Setup

The FY '09 Test Plan defines the following Flight Hazards service composition setup:

1. Uses MIT "NNEW Federation 1" Registry.
2. Targets NCAR Datasets: CEIL, CIP-20, CIPSEV-20, FLTCAT, GTG2, VIS, NCWD served by NCAR WCS.
3. Threshold Decision Service. Identifies threshold values for each hazard variable by running aircraft and pilot characteristics through a rules engine. Deployed at NCAR.
4. Data Thresholding Service. Reduces continuously variable values of hazards into discrete bins according to the thresholds provided. Results in a dataset with a small number of separate values. Deployed at NCAR.
5. Corridor Image Generator Service. A service to portray vertical and horizontal views.

7.2 Verification of 4D Trajectory Capabilities and Flight Hazard Service Composition with Flight Hazard Tool Web UI

These tests demonstrate the trajectory capabilities of WCS, the validation of the trajectory specification and the atomic services composition into a higher level of Decision Support System. We are going to run two tests, an historic use case that targets archived data so we are able to reproduce exactly the expected results for this test plan and one case that will be driven by the tester's choices to test the tool targeting forecast data.

7.2.1 Weather Hazards Along a Flight Trajectory - Archived Use Case

This test demonstrates the derivation of weather hazards from several data coverage datasets given a flight plan containing a trajectory created with the user provided inputs targeting archived data.

1. In a web browser, launch the Flight Wx Hazard Tool:

<http://weather.aero/nnew/fy09/fwht>

Use “**nnew**” and “**TBO4all**” for the login credentials.

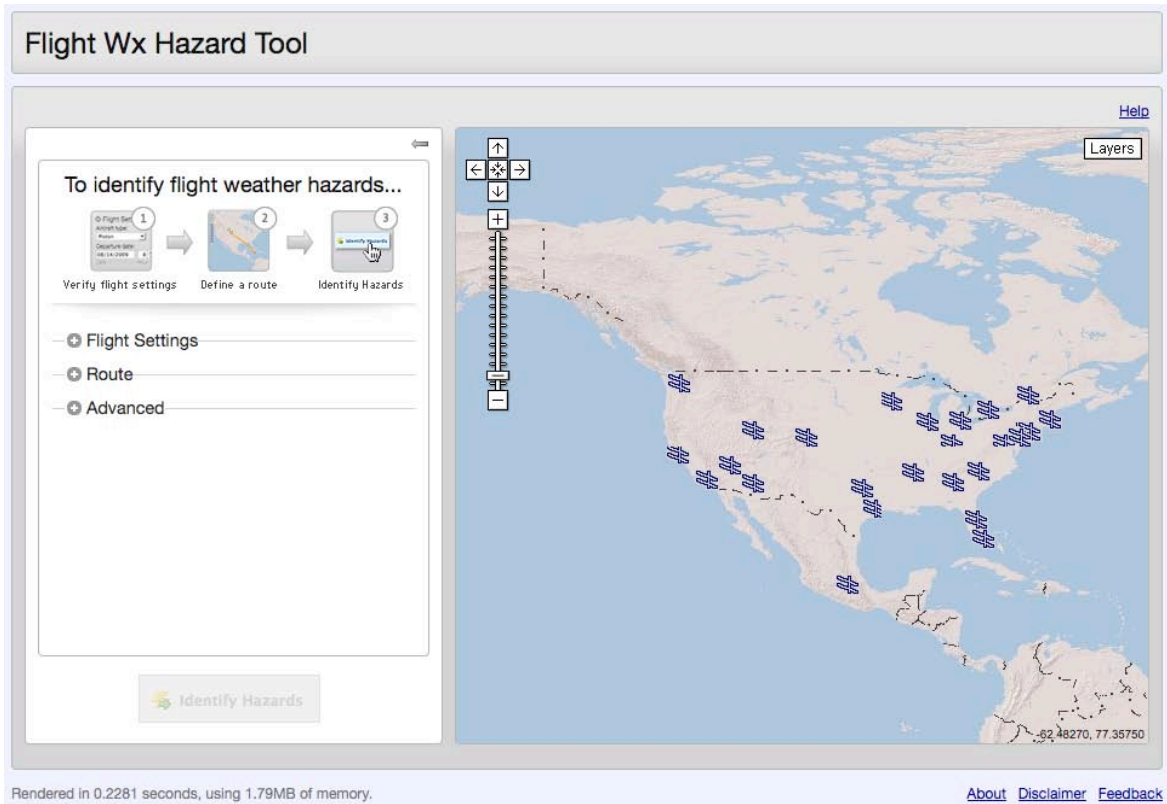


Figure 7.1 Flight Wx Hazard Tool User Interface

2. Click on “**Flight Settings**” label to verify flight settings:

Flight Settings

Aircraft type:

Flight type: VFR IFR

Departure date: UTC
Date (mm/dd/yyyy) Hour Minute

Cruise altitude: feet

Cruise speed: knots

Ascent rate: ft/min

Descent rate: ft/min

Ascent speed: knots

Descent speed: knots

Figure 7.2 Flight Settings Form

3. Set **"Aircraft type"** to **"Turboprop"**.
4. Set **"Departure date"** to **"09/09/2009 17:00"**.
5. Click again on **"Flight Settings"** label to collapse the form and click on the **"Route"** label to define a route.
6. Type the following waypoints into the text field:
 - Baltimore - click enter to add first displayed option **"KBWT"**
 - Hinch Mountain - click enter to add first displayed option **"HCH"**
 - OKW - click enter to add first displayed option **"OKW"**
 - Southeast Texas Rgn1 - click enter to add first displayed option **"KBPT"**

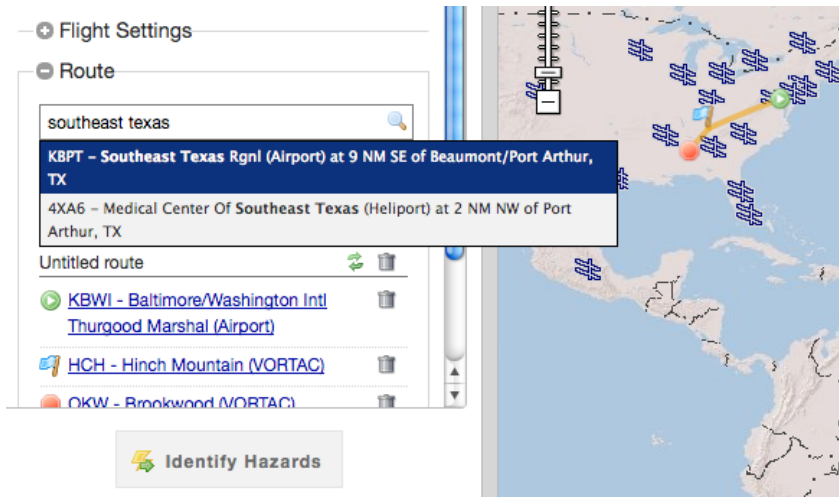


Figure 7.3 Route Form

7. Click the “**Identify Hazards**” button at the bottom. The tool submits a request to the Flight Hazard Services and returns a response in about 8 to 10 seconds. The response time is higher for archived cases than for forecast cases due to the higher amount of available data.

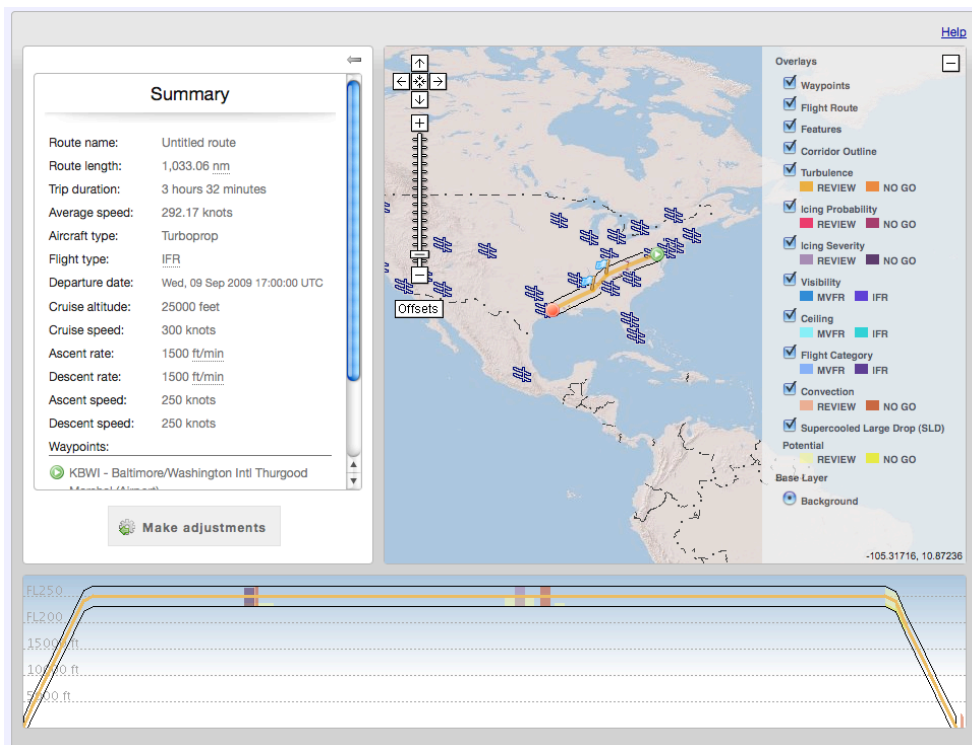


Figure 7.4 Hazards View

The tool displays a summary of the user Flight Plan on the left and displays the derived weather hazards along the trajectory in the map and in a profile view of the trajectory at the bottom. Also a legend with the colors used to identify different types the hazards is generated on the right.

8. Double click twice on the map over the middle of the flight trajectory to zoom in to identify hazards on the corridor. Click also on the arrow on top of the summary to expand the map over the left.

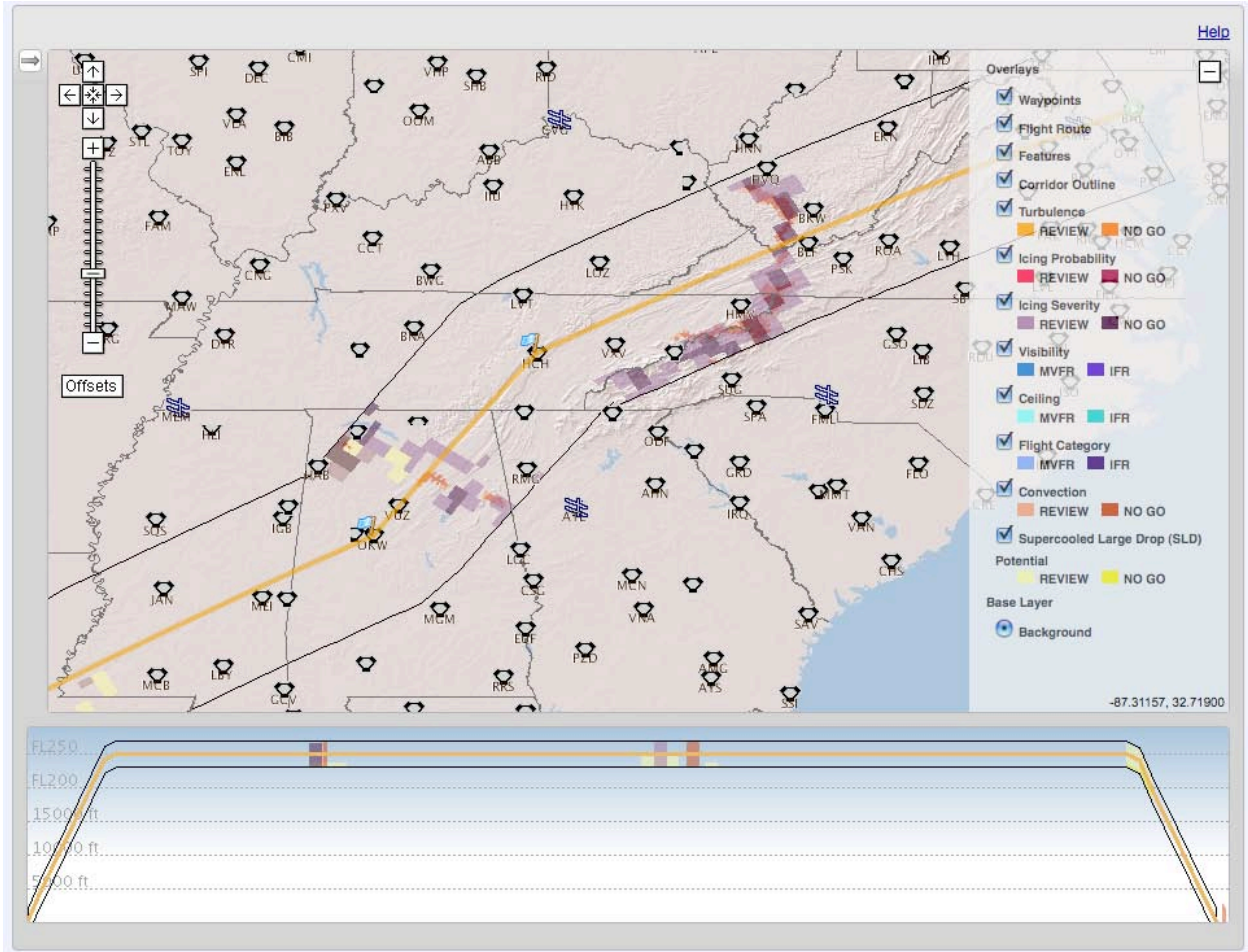


Figure 7.5 Weather Hazards Map and Profile View

We can identify Icing Severity, Super Large Drop Icing and Convection hazards along the corridor close to HCH and OKW waypoints.

9. Click on map and pan to the right until the landing KBPT airport (red icon) shows in the map. Double click twice to zoom in to identify hazards.

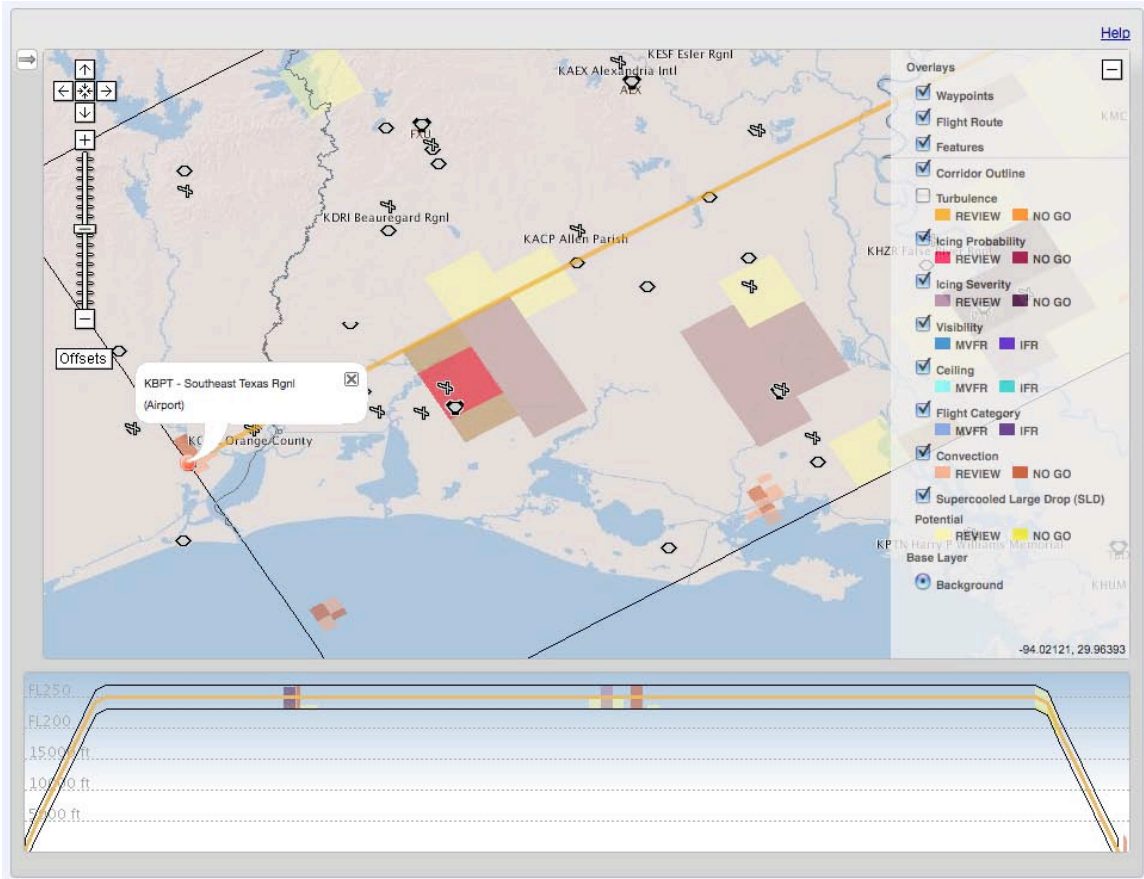


Figure 7.6 Weather Hazards Map and Profile View

We can identify Icing Severity, Icing Probability and Super Large Drop Icing and hazards along the corridor close to KBPT landing airport.

10. Enable/disable hazard layers clicking on the check boxes on the layers/legends panel to identify single types of hazards.

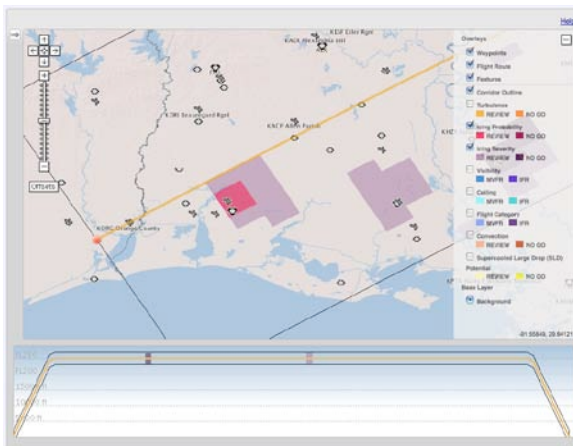


Figure 7.7 Icing Severity and Probability Hazards

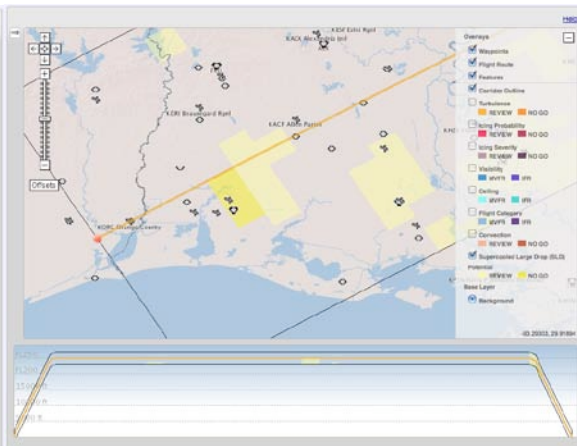


Figure 7.8 Icing SLD Hazards

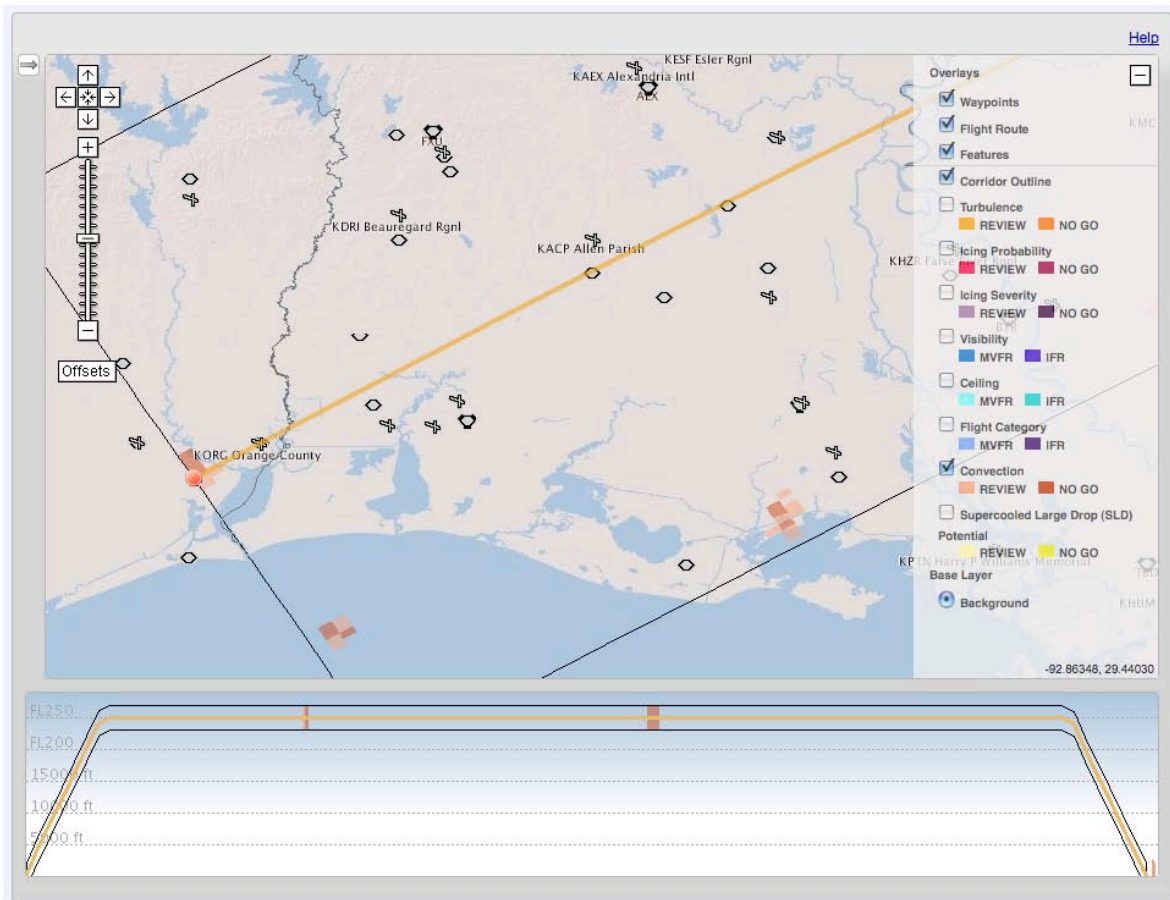


Figure 7.9 Convection Hazards

11. Turn on all the hazard layers. Click on the **“offsets”** button under the map navigation controls. Offsets refer to deviations from the centerline of the flight corridor. Adjusting these values allows you to look around within the corridor. Type **“1500”** feet on the Z-Offset input box and click somewhere on the interface so the change takes place. This triggers a request to display a horizontal cross section of the corridor 1500 feet over the centerline of the trajectory. We can observe that at this altitude for the same trajectory there is no icing probability hazard displayed.

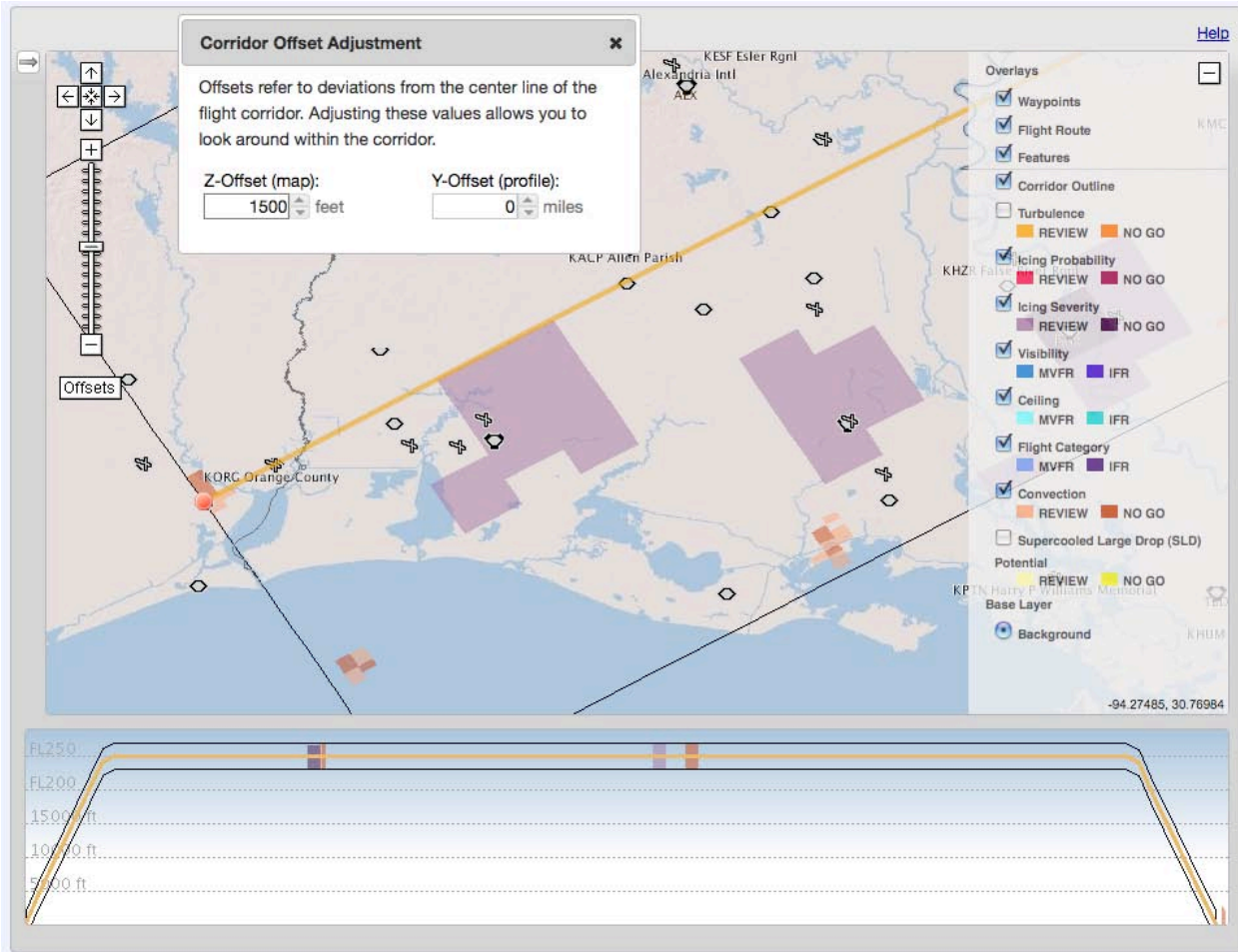


Figure 7.10 Z Offset

12. Spin down on the Y-Offset (profile) to “-15” miles. This triggers a request to display a vertical cross section of the corridor -15 miles over the centerline of the trajectory. We can observe on the profile view there are new Icing severity and probability hazards displayed on the right side where the plane starts it’s descending.

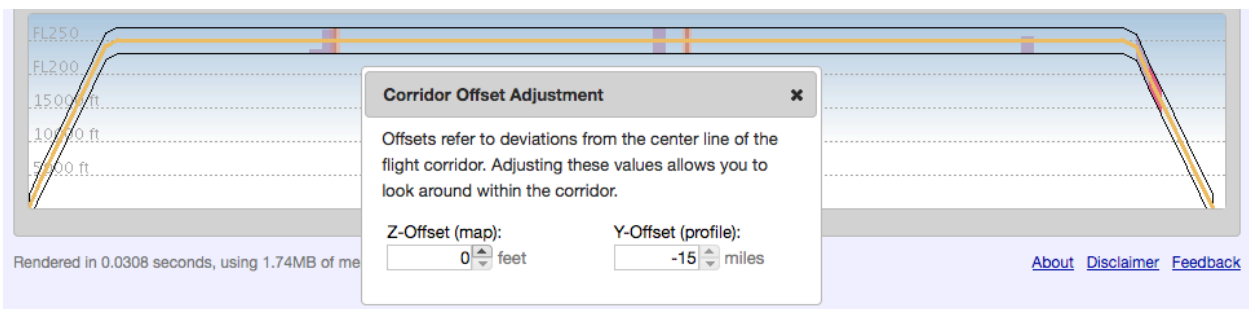


Figure 7.11 Y Offset

13. Click the arrow at the top-left of the tool to get back to the summary view. Click “**Make Adjustments**” to redefine some of the flight plan parameters.

14. Click on “**Flight Settings**”. Select “**Piston**” and check that Flight Type is set to “**VFR**” instead
15. Click the “**Identify Hazards**” button
16. Using map navigation controls zoom out

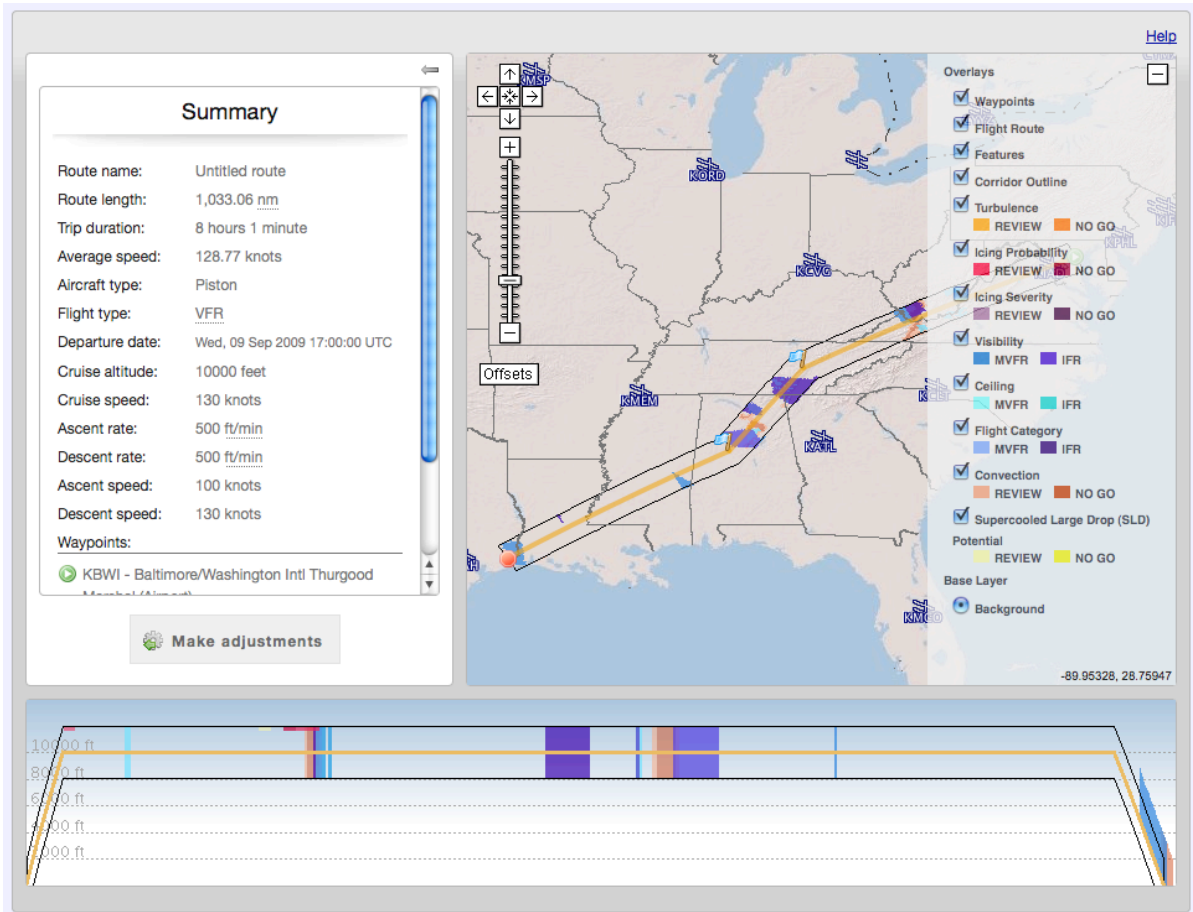


Figure 7.12 Modified Hazards

We can observe that a total different set of hazards is displayed for this flight plan that contains the same trajectory but different aircraft and flight characteristics. We can observe several MVFR and IFR flight conditions along the trajectory.

7.2.2 Weather Hazards Along a Flight Trajectory – “Near Future” Use Case

This test demonstrates the derivation of weather hazards from several data coverage datasets given a flight plan containing a trajectory created with the user provided inputs. This test will target mostly forecast data so the results cannot be predictable in advance.

1. Click on “**Flight Settings**.”
2. Select an “**Aircraft type**.” Feel free to change any other parameter also.
3. Set “**Departure date**” to a date in the future no further than 4 hours from now. Most of the datasets does not go more than 12 hours in the future.

4. Click again on the “**Flight Settings**” label to collapse the form and click on the “**Route**” label to define a Route.
5. Define a route typing the desired waypoints on the input text.
6. Click the “**Identify Hazards**” button at the bottom.
7. Identify hazards visually on the map and profile view. Zoom in on map if it’s necessary.

7.3 Verification of Flight Hazard Composition Service and Composite Services with SoapUI

SoapUI is an application that provides viewing and invocation of SOAP XML requests towards web services, as well as viewing of SOAP XML web services responses. Using these capabilities and some additional ones that allow the transfer and creation of properties between different services responses and requests we created test suits and test cases for the Flight Hazards Service and the services part of the composition.

The Flight Hazard web service itself is an orchestrator service which main function is to orchestrate several atomic services building service requests and adapting service responses and combine them into new service requests for other services. The Flight Hazard Service orchestrates the MIT’s ebXML Registry, WCS, a Thresholds Decision Service and a Data Thresholding Service. We exposed all this service through a WSDL interface including the Flight Hazards Service.

In the test cases created using SoapUI we want to test and demonstrate each individual service, the composition/orchestration of this services and the Flight Hazards Service as a black box.

1. Launch SoapUI 2.5.1 on a **Linux** machine.

Note: we will be using version 2.5.1 of the tool since some features required for some of the tests are broken in version 3.0 and 3.0.1 of the tool.

2. Click “**File**” → “**Import Remote Project**” and type:

http://weather.aero/nnew/fy09/fwht/FlightHazardsService_Composite-soapui-project.xml

Click “**OK**” and then type “**nnew**” and “**TBO4all**” for the credentials.

Note: If, for some reason, this does not work, save the file locally and import the project from the file system a pop up window will appear, warning about a missing attachment. This is an expected message. Click “**OK**” and “**Yes**” to the “**There are unresolved paths, continue?**” dialog.

7.3.1 Atomic Services Test Steps

We will test first each service part of the orchestration individually submitting a soap request and doing some assertions on the response. We setup the services requests with values to simulate the same historic use case we saw from the Flight Hazard Tool.

1. Expand “FlightHazardsService_Composite” project on the “Navigator” panel if it is not expanded already.
2. Expand “FlightHazard Composition Services TestSuite” node and double click on “Composition TestCase”.

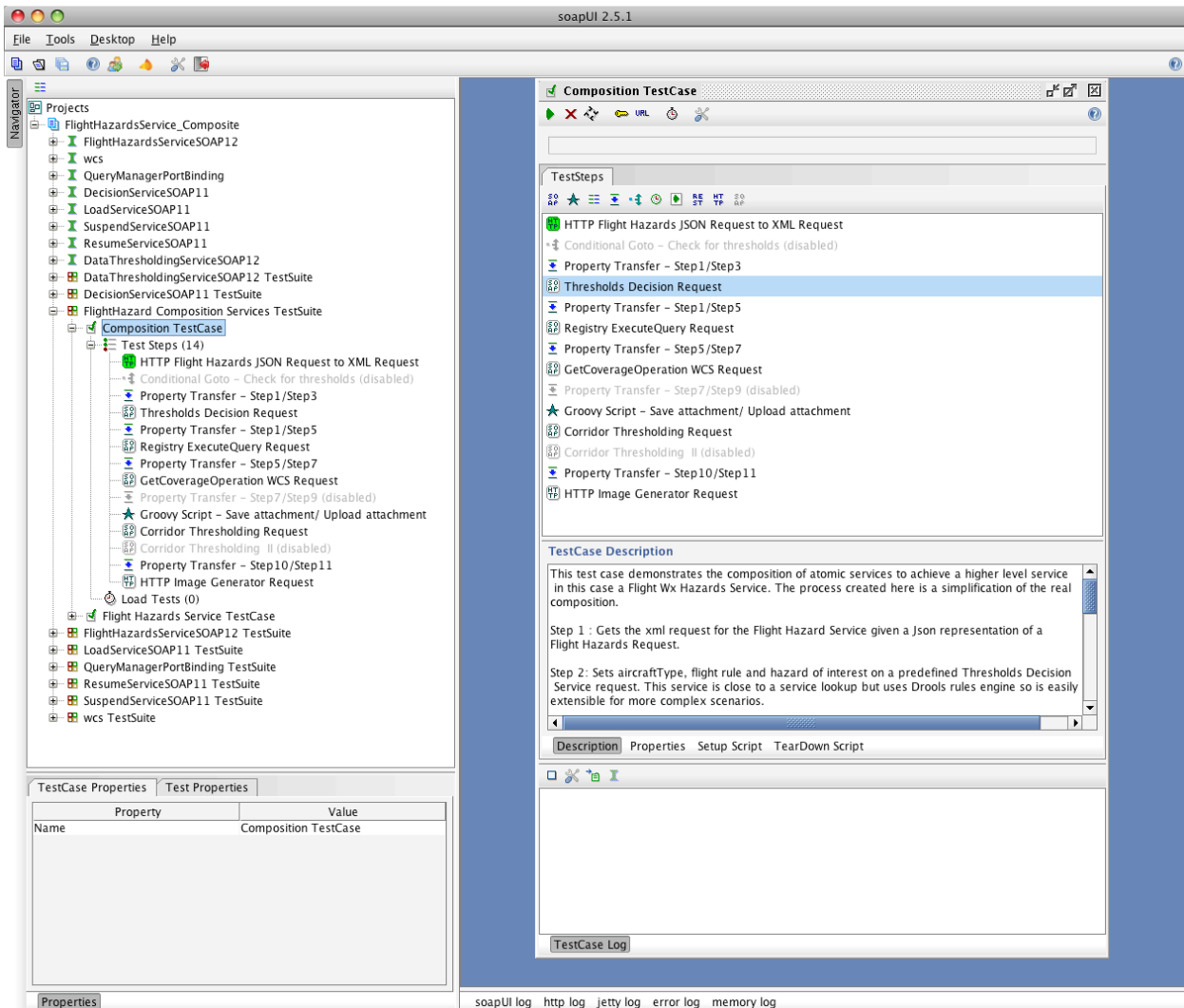


Figure 7.13 Flight Hazard Service Test Suite

3. Expand the new opened window “Composition TestCase” until all the “Tests Steps” are visible. NOTE: ALL OF THE STEPS IN THE TEST SUITE MUST BE RUN INDIVIDUALLY. There is a bug in SoapUI which will cause it to hang if you attempt to run the entire suite at once.
4. Double click on “HTTP Flight Hazards JSON Request to XML Request”.

This test step is a JSON request equivalent to the one the Flight Hazard Tool UI generates. It sends the JSON request to a proxy service that transforms it into a valid XML request for the Flight Hazard Service

- Click the “**submit request**” icon on the left-top corner. If all assertions on the received response are passed a green icon is displayed on the top-left corner of the window.

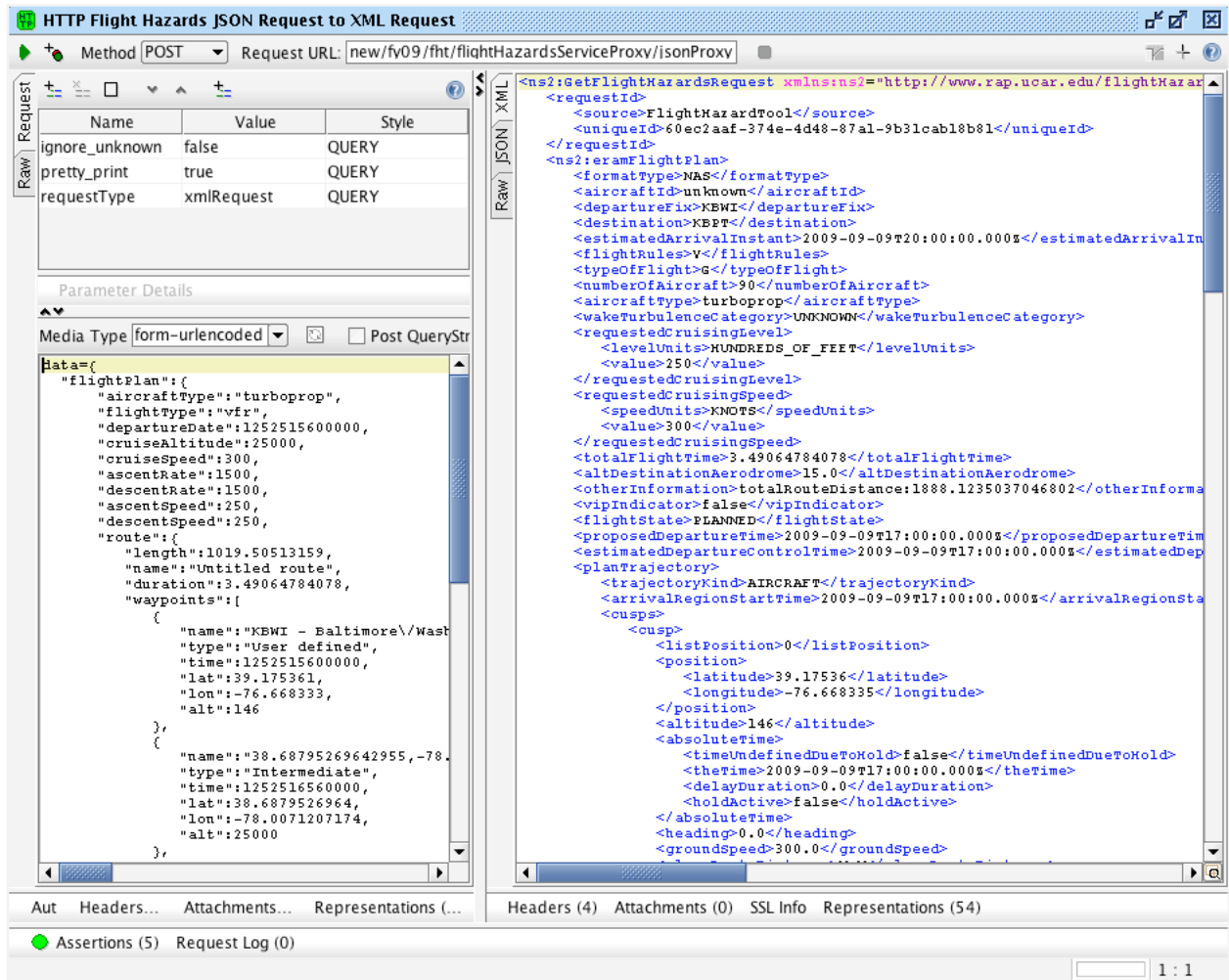


Figure 7.14 Successful Test

- “Close” the window prior to running the next test.
- Double click on “**Thresholds Decision Request.**”

This test sends the thresholds request for a hazard of interest in this case for Icing Severity and for aircraft type Turboprop and VFR flight rule and gets back a response that contains a set of thresholds.

- Click the “**submit request**” icon on the left-top corner. If all assertions on the received response are passed a green icon is displayed on the top-left corner of the window. Click on the “**Assertions**” button/label to display the passed assertions.

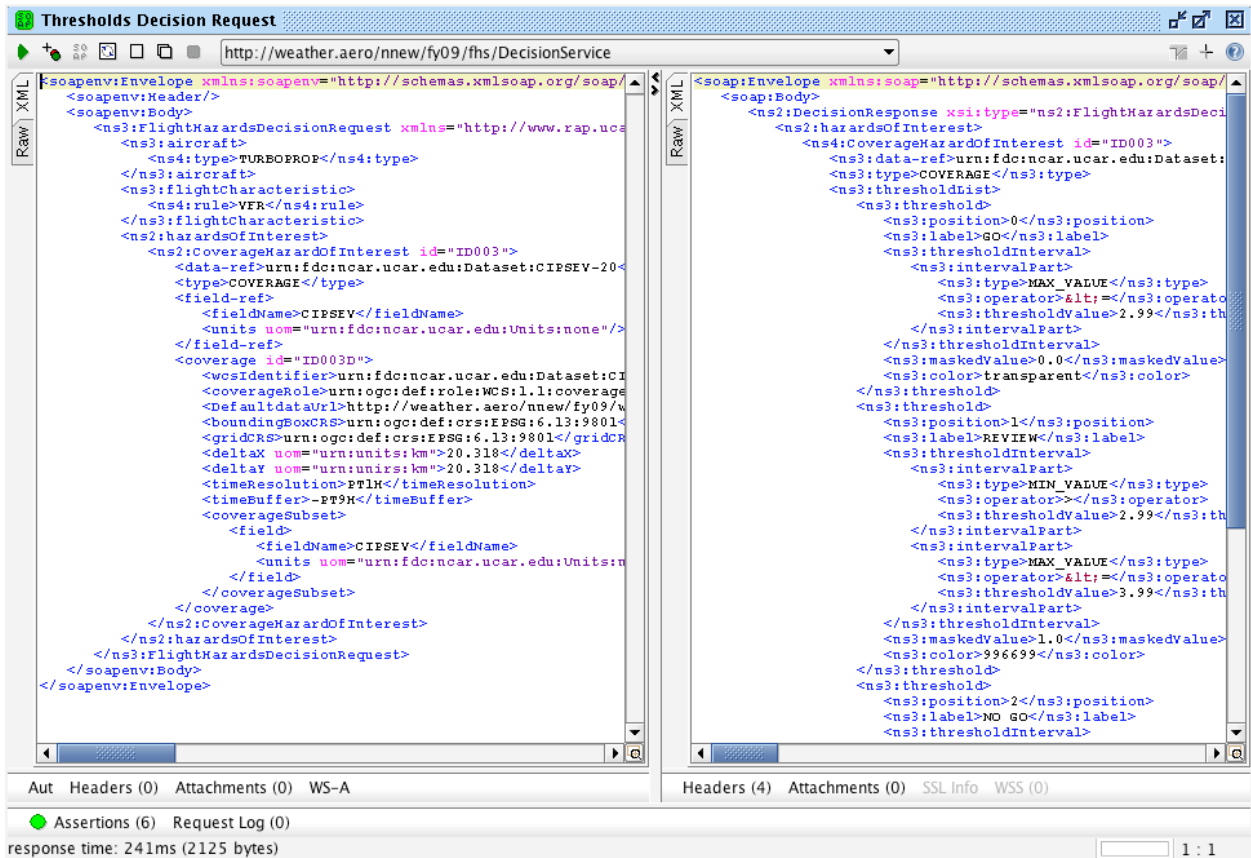


Figure 7.15 Passed Assertions

9. “Close” the window prior to running the next test.
10. Double click on “Registry ExecuteQuery Request.”

This test sends a registry query request to the MIT registry to get an endpoint to get the data for the hazard of interest.

11. Click the “submit request” icon on the left-top corner. If all assertions on the received response are passed a green icon is displayed on the top-left corner of the window. Click on the “Assertions” button/label to display the passed assertions.

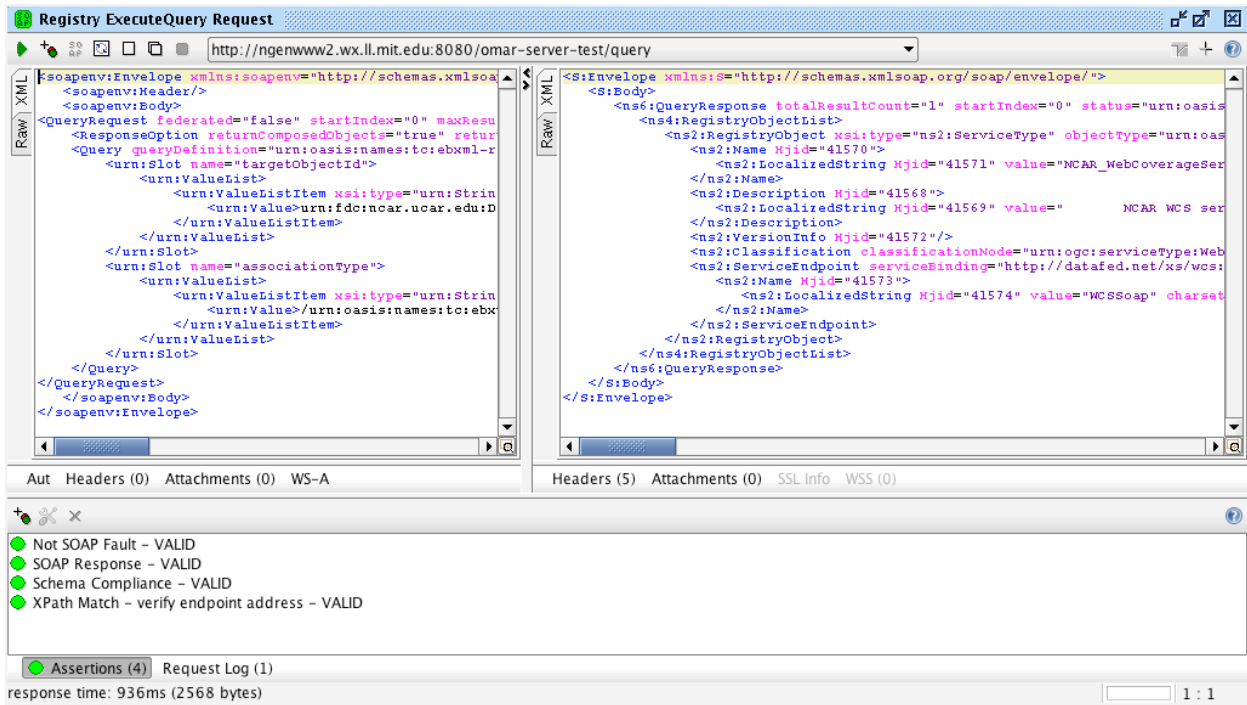


Figure 7.16 Assertions

12. “Close” the window prior to running the next test.

13. Double click on “GetCoverageOperation WCS Request.”

This test sends a predefined WCS corridor request to endpoint returned by the registry. In the real implementation, the request is created from the Flight Hazards request provided data, concretely from the ERAM flight plan and trajectory waypoints to create a WCS corridor request.

14. Click the “submit request” icon on the left-top corner. If all assertions on the received response are passed a green icon is displayed on the top-left corner of the window. Click on the “Assertions” button/label to display the passed assertions.

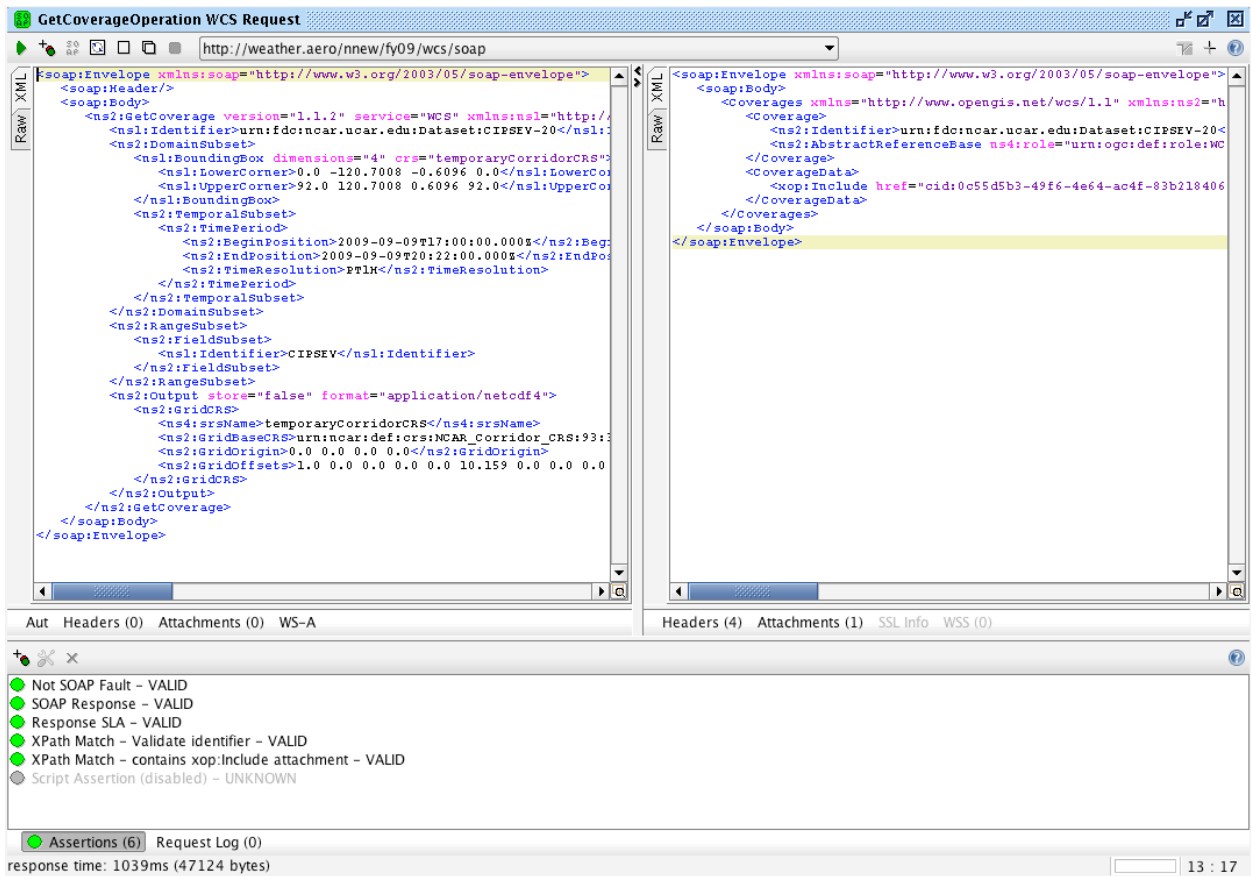


Figure 7.17 GetCoverage Test

15. “Close” the window prior to running the next test.

16. Double click on “Corridor Thresholding Request.”

This test sends a predefined Data Thresholding Service request with no corridor data attached initially. In the FHS implementation the thresholds values are set from the previous Thresholds Decision Service response.

17. Click the “submit request” icon on the left-top corner. The test step request **fails** as it is expected. Click on “Raw” tab on the request pane. Notice that the request provided does not have netcdf corridor data attached. The file provided for attachment does not exist on the local file system yet, so the service returns an invalid response in terms of schema (since it is missing the location of the thresholded corridor data).

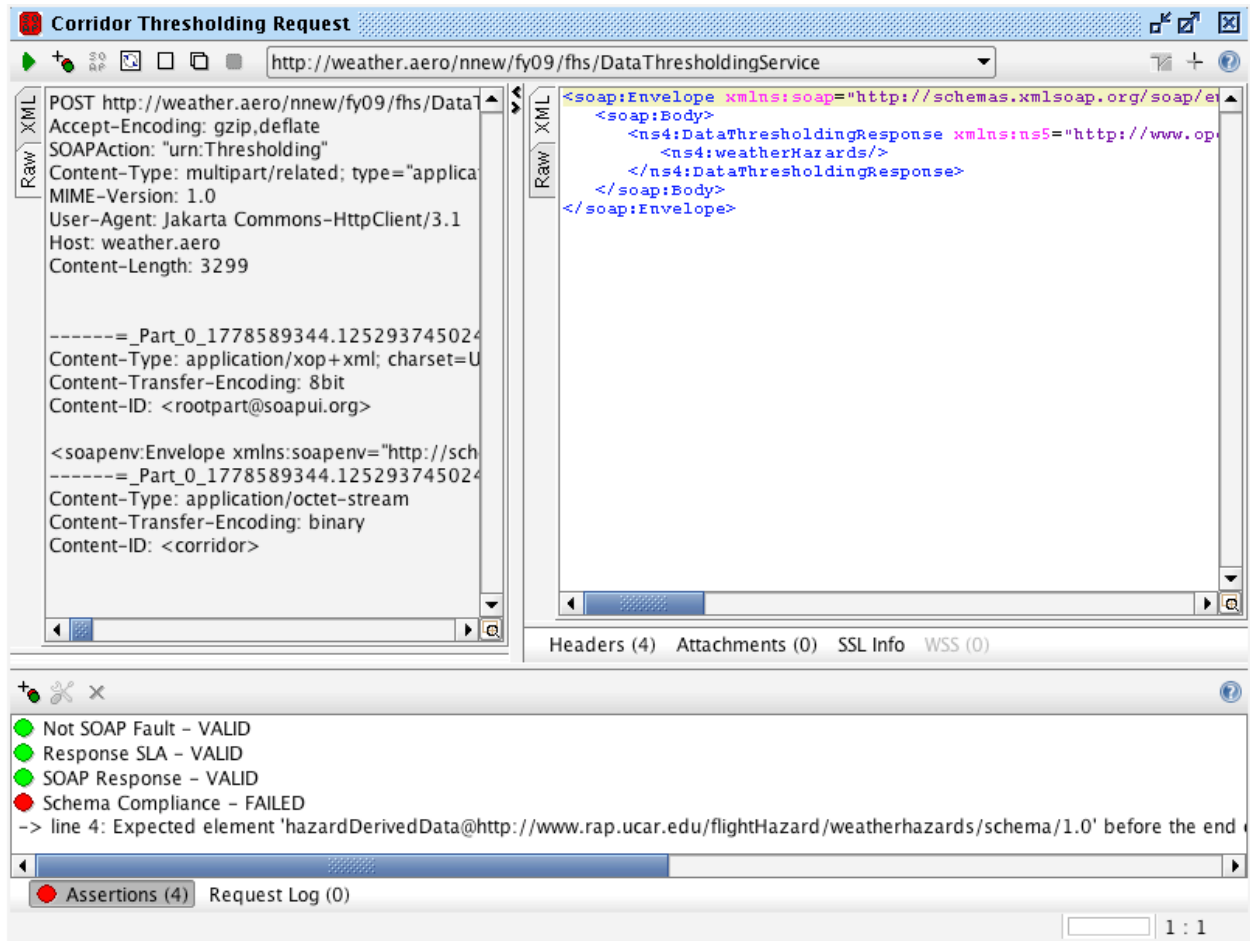


Figure 7.18 Failed Test

18. To run the test successfully, double click on the “**Groovy Script - Save attachment/ Upload attachment**” test step above the “**Corridor Thresholding Request.**”

This test script step saves the attachment containing the corridor data from the previous WCS response and attaches it to the Thresholding Service request.

19. Click the “**Run script**” icon on the left-top corner. Check the logs after running the script in green at the bottom of the window.

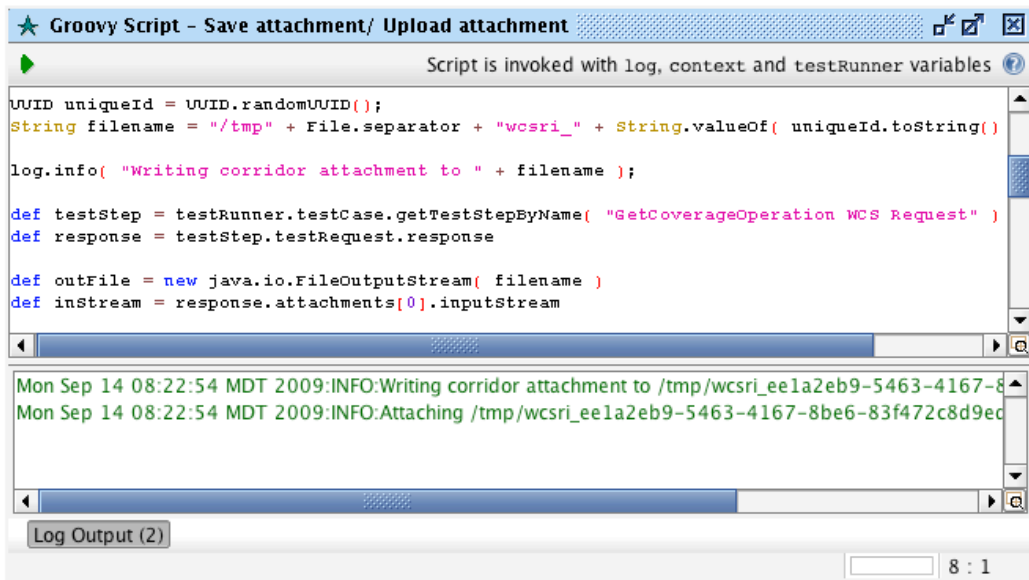


Figure 7.19 Groovy Script

20. Click the “submit request” icon on the left-top corner of the “Corridor Thresholding Request” window again. This time we can see the netcdf corridor data attached on the “Raw” tab of the request panel. The response now contains the location for the corridor thresholded data and the thresholds applied to it. If all assertions on the received response are passed a green icon is displayed on the top-left corner of the window. Click on the “Assertions” button/label to display the passed assertions.

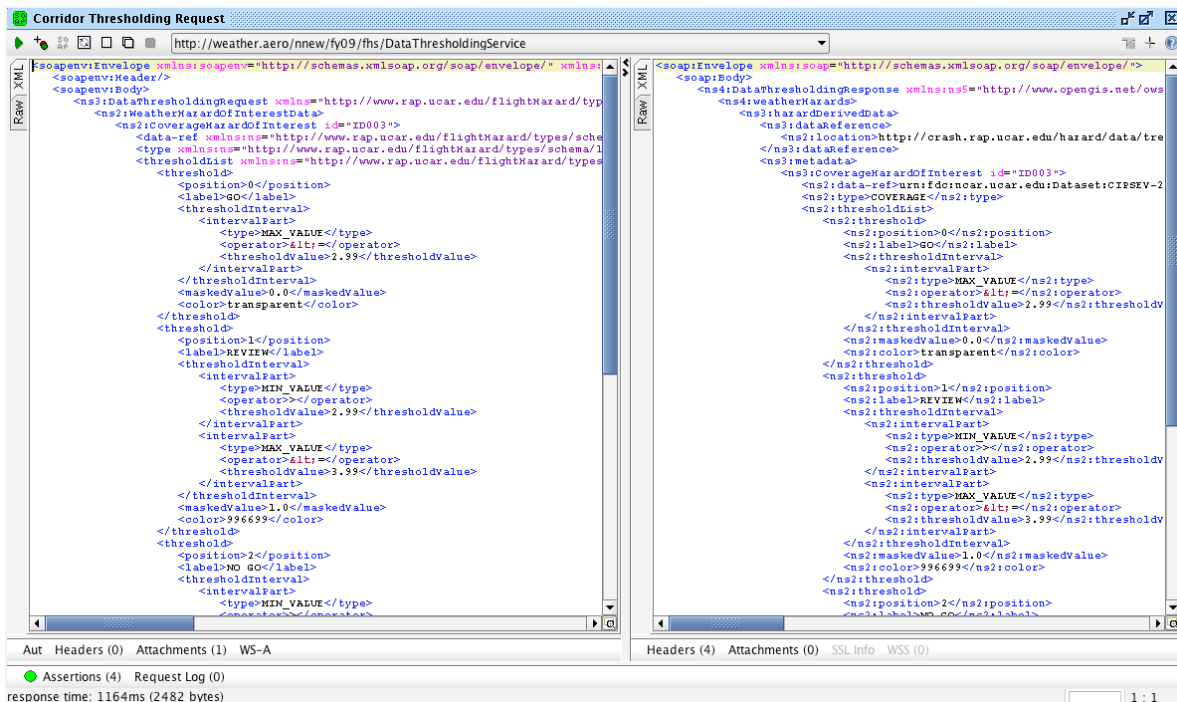
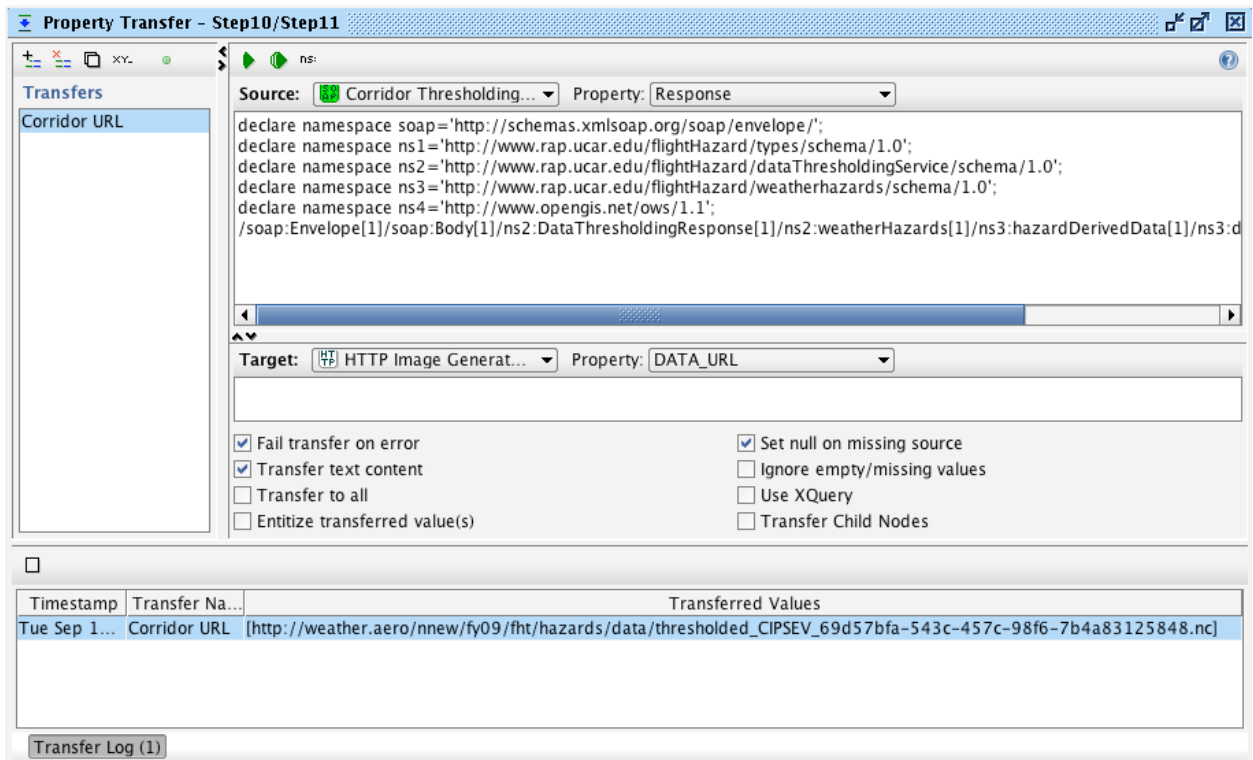


Figure 7.20 Successful Test

21. “Close” the window prior to running the next test.
22. Double click on the “Property Transfer - Step10/Step11” test step. Click on green icon “Run selected property transfer”.

This step sets the URL location of the thresholded corridor data returned by the Thresholding Service response into the DATA_URL parameter for the Image Generator Service.

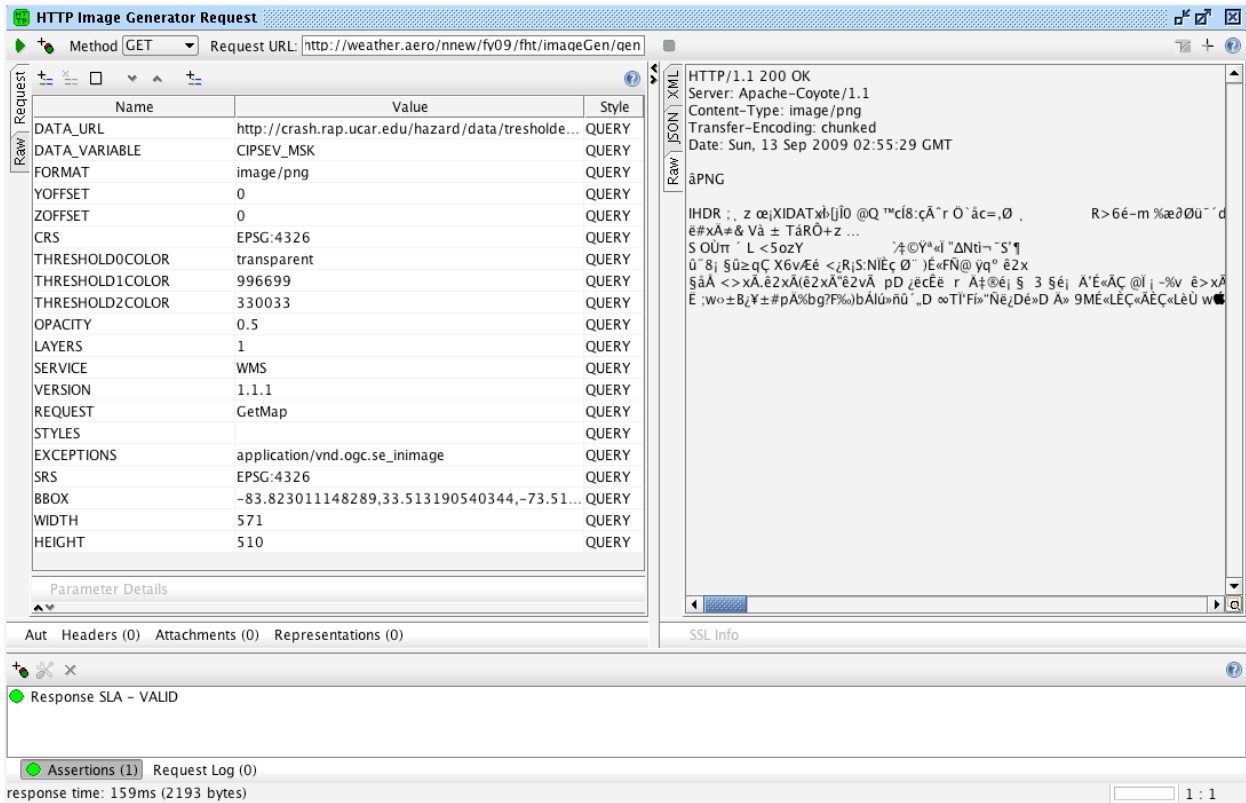


23. Double click on “HTTP Image Generator Request.”

This test sends a request to the corridor hazards image generator with the URL where the thresholded data is stored as a query parameter. It simulates the request sent by the FHT user interface send after receiving the response from FHS to the image generator service. The image generator returns a PNG image that depicts the weather hazards. Soap UI just displays the binary content of the PNG image.

http://weather.aero/nnew/fy09/fht/imageGen/gen?DATA_URL=http://weather.aero/nnew/fy09/fht/hazards/data/thresholded_CIPSEV_7fea9239-483d-45ac-a3b1-0db9862ba7d5.nc&DATA_VARIABLE=CIPSEV_MSK&FORMAT=image/png&YOFFSET=0&ZOFFSET=0&CRS=EPSG:4326&THRESHOLD0COLOR=transparent&THRESHOLD1COLOR=996699&THRESHOLD2COLOR=330033&OPACITY=0.5&LAYERS=1&SERVICE=WMS&VERSION=1.1.1&REQUEST=GetMap&STYLES=&EXCEPTIONS=application/vnd.ogc.se_inimage&SRS=EPSG:4326&BBOX=-83.823011148289,33.513190540344,-73.514802944663,42.720171597872&WIDTH=571&HEIGHT=510

- Click the “**submit request**” icon on the left-top corner. If all assertions on the received response are passed a green icon is displayed on the top-left corner of the window. Click on the “**Assertions**” button/label to display the passed assertions. Click on the “**Raw**” tab on the response panel to see detailed info about the response.



- “Close” the window prior to running the next test.

7.3.2 Flight Hazard Service Composition Test Case

This test demonstrates the composition of the atomic services to achieve a higher-level service through composition and orchestration. The test case tries to simulate the FHS implementation for one single hazard of interest, Icing Severity.

- On the “**Composition TestCase**” window click on the icon on the top-left corner to run the test case. The test case runs the requests tested previously in sequence order but uses the transfer steps (XPATH/XQUERY and Groovy scripts) in between to set properties on the request from previous responses. The final result on the last request of the orchestration test case is a PNG image representing some of the icing hazards along the flight trajectory given a JSON representation of a flight plan and the hazard of interest.

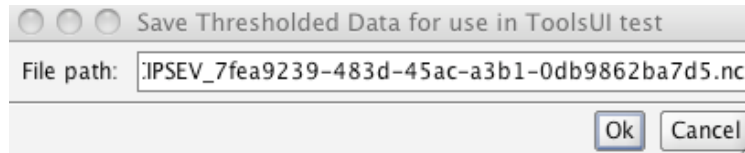


Figure 7.21 Save Data

Note that you should save this file path for use in 7.4 step 3

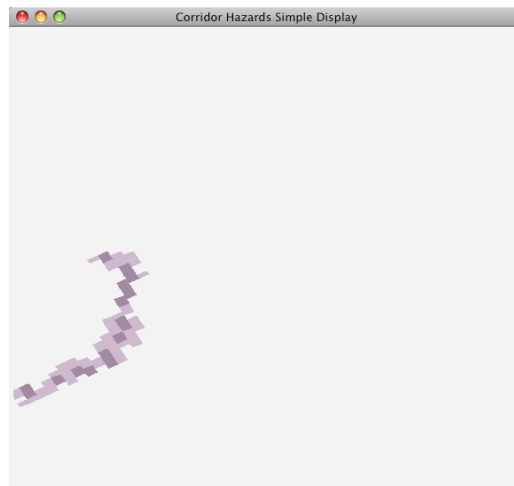


Figure 7.22 Image Generation Service Output

Compare the Icing Severity Hazards displayed from the image generator service with the same case requested from the Flight Hazard Tool User Interface.

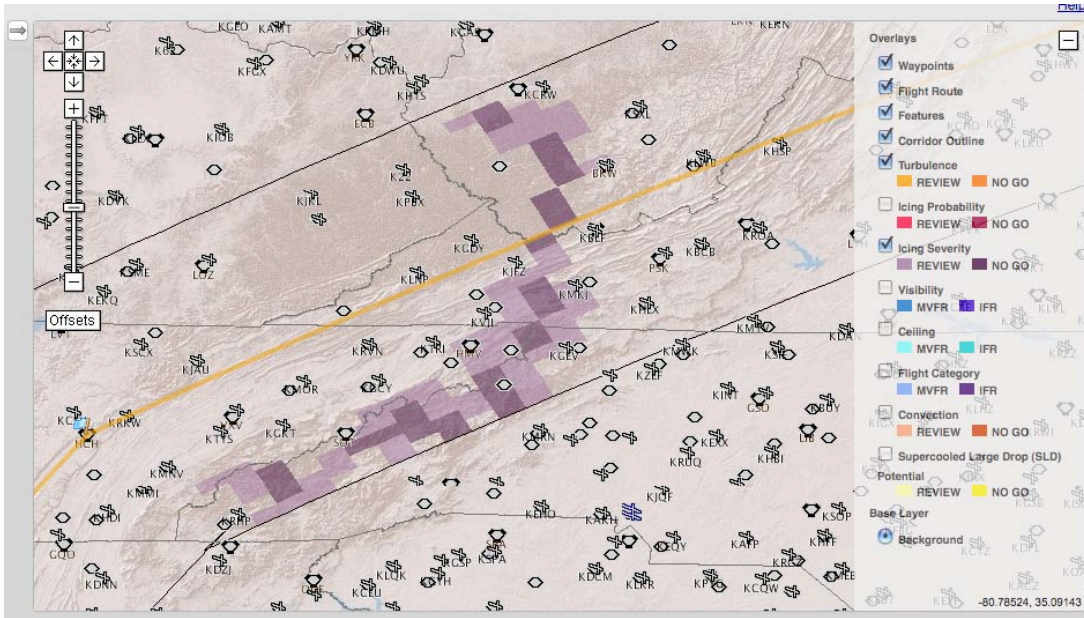


Figure 7.23 Flight Hazard Tool Output

The test case also pops up a dialog box after the Thresholding Service request has ended. Save the Thresholded data from the URL embedded in the xml response to the local file system. It will be used in the next verification ToolsUI test to verify the thresholded corridor data contents.

The “**Composition TestCase**” window should look like the image below after running the test case.

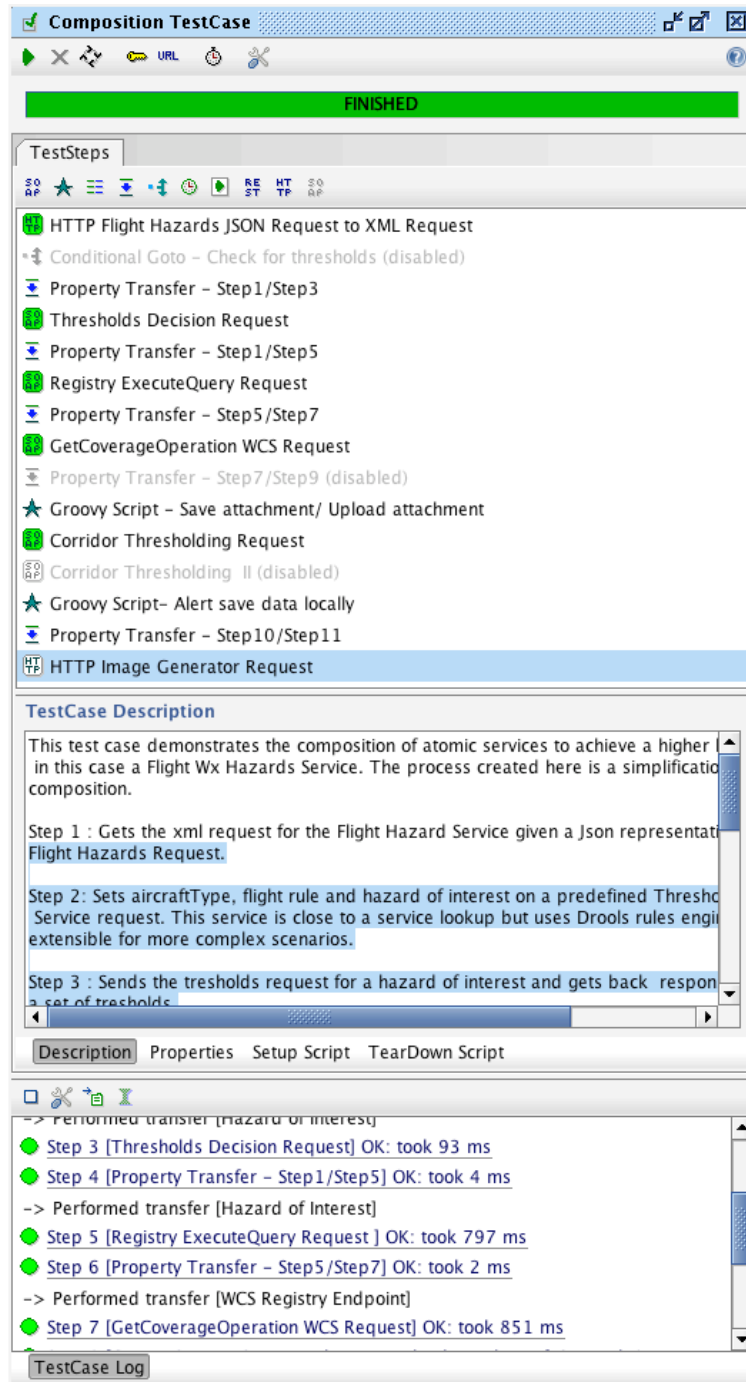


Figure 7.24 Composition Test Case

“Close” the window prior to running the next test.

7.3.3 Flight Hazards Service – Black Box Test Case

This test case runs black box Flight Hazard Requests. The test case is composed by two tests. The first test runs a JSON Flight Hazard Request for multiple hazard of interest and the second one runs an XML Flight Hazard Request for multiple hazards of interest.

1. Double click on “**Flight Hazards Service - Black Box TestCase**” under “**FlightHazard Composition Services TestSuite**” on the “**Navigator**” panel.
2. Click on the icon on the top-left corner to run the test case.

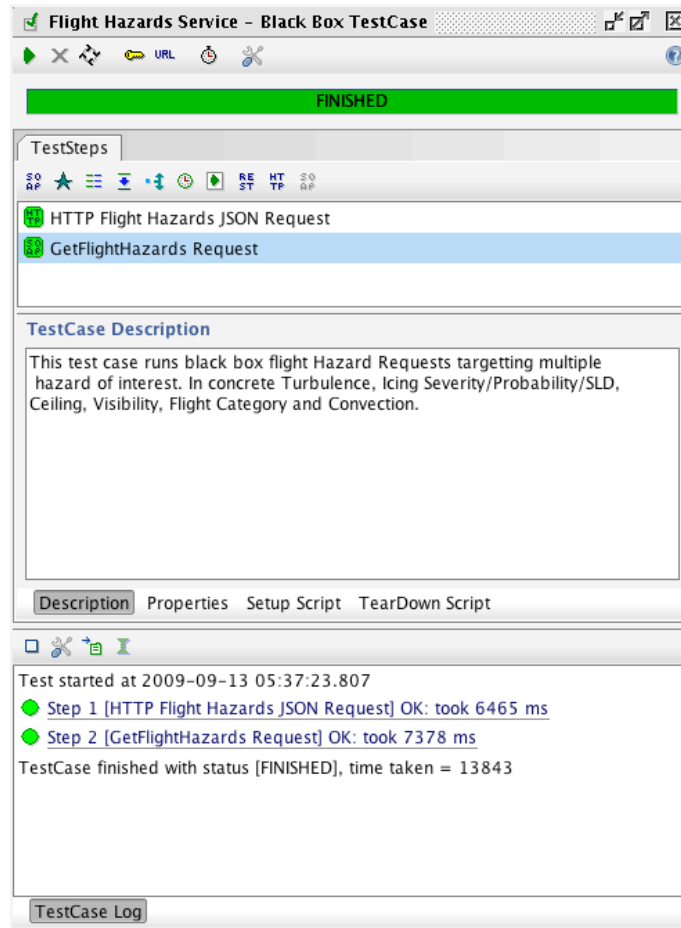
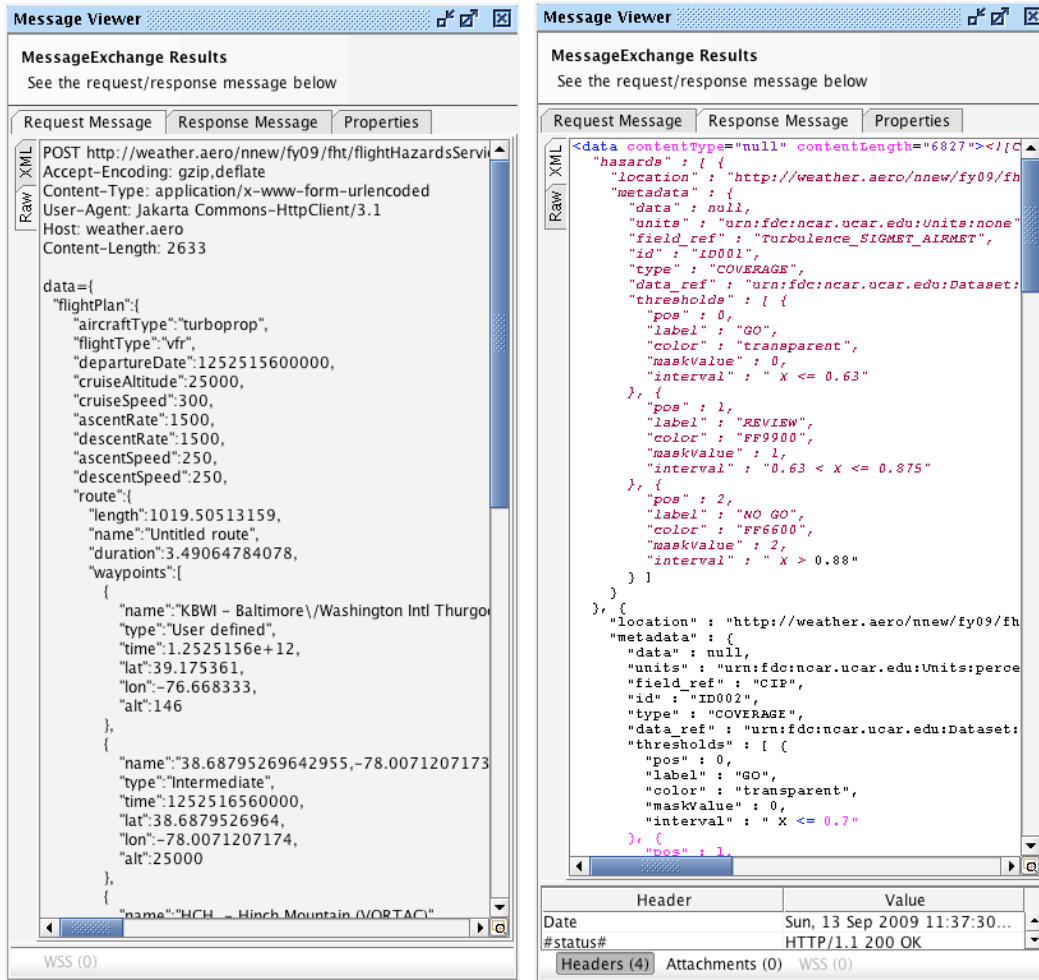


Figure 7.25 Black Box Test Case

3. Double click on the “**Step 1**” and “**Step 2**” links at the bottom of the window to check the Request Message and Response message for each test.



7.4 Verification Derived Data with ToolsUI

1. Launch ToolsUI from <http://www.unidata.ucar.edu/software/netcdf-java/v4.0/webstart/netCDFtools.jnlp>
2. Select the “NCDump” tab.
3. In the “**command**” input box type the path of the previous downloaded corridor thresholded file and press **Enter**.

Reminder: This file should be located in the **/tmp** directory if it was not changed.


```

command: /tmp/CIPSEV_ec4475dd-ee93-4efe-950e-07f3b0a46d93.nc
netcdf /tmp/CIPSEV_ec4475dd-ee93-4efe-950e-07f3b0a46d93.nc {
  dimensions:
    z_dim = 13;
    y_dim = 25;
    x_dim = 93;
  variables:
    int CIPSEV_MSK(z_dim=13, y_dim=25, x_dim=93);
      :long_name = "Thresholded CIPSEV";
      :standard_name = "CIPSEV_MSK";
      :threshold0 = "GO:transparent:0:MAX_VALUE=<=2.99";
      :threshold1 = "REVIEW:996699:1:MIN_VALUE=>2.99:MAX_VALUE=<=3.99";
      :threshold2 = "NO GO:330033:2:MIN_VALUE=>3.99";
    double time(x_dim=93);
      :_FillValue = NaN; // double
      :long_name = "time";
      :standard_name = "time";
      :units = "ms";
    float longitude(y_dim=25, x_dim=93);
      :long_name = "longitude";
      :standard_name = "longitude";
      :units = "degrees";
      :_FillValue = NaNf; // float
    float latitude(y_dim=25, x_dim=93);
      :long_name = "latitude";
      :standard_name = "latitude";
      :units = "degrees";
      :_FillValue = NaNf; // float
    float altitude(z_dim=13, x_dim=93);
      :long_name = "altitude";
      :standard_name = "altitude";
      :units = "km";
      :_FillValue = NaNf; // float
    float CIPSEV(z_dim=13, y_dim=25, x_dim=93);
      :standard_name = "CIPSEV";
      :units = "index";
      :_FillValue = NaNf; // float
      :long_name = "CIP Severity ";

:FileFormat = "Netcdf4";
:Conventions = "CF-1.4";

```

Figure 7.26 ToolsUI Verification

The thresholded corridor dataset contains the original corridor data (CIPSEV) and the thresholded data (CIPSEV_MSK) . For this dataset the dimensions are 93 sample points along the flight trajectory on the X dimension, 25 on the Y dimension of the corridor (width) and 13 on the Z dimension of the corridor (height)

4. Add “-v CIPSEV” to the command after the dataset filename and press **Enter** to dump the CIPSEV variable containing the original corridor data.

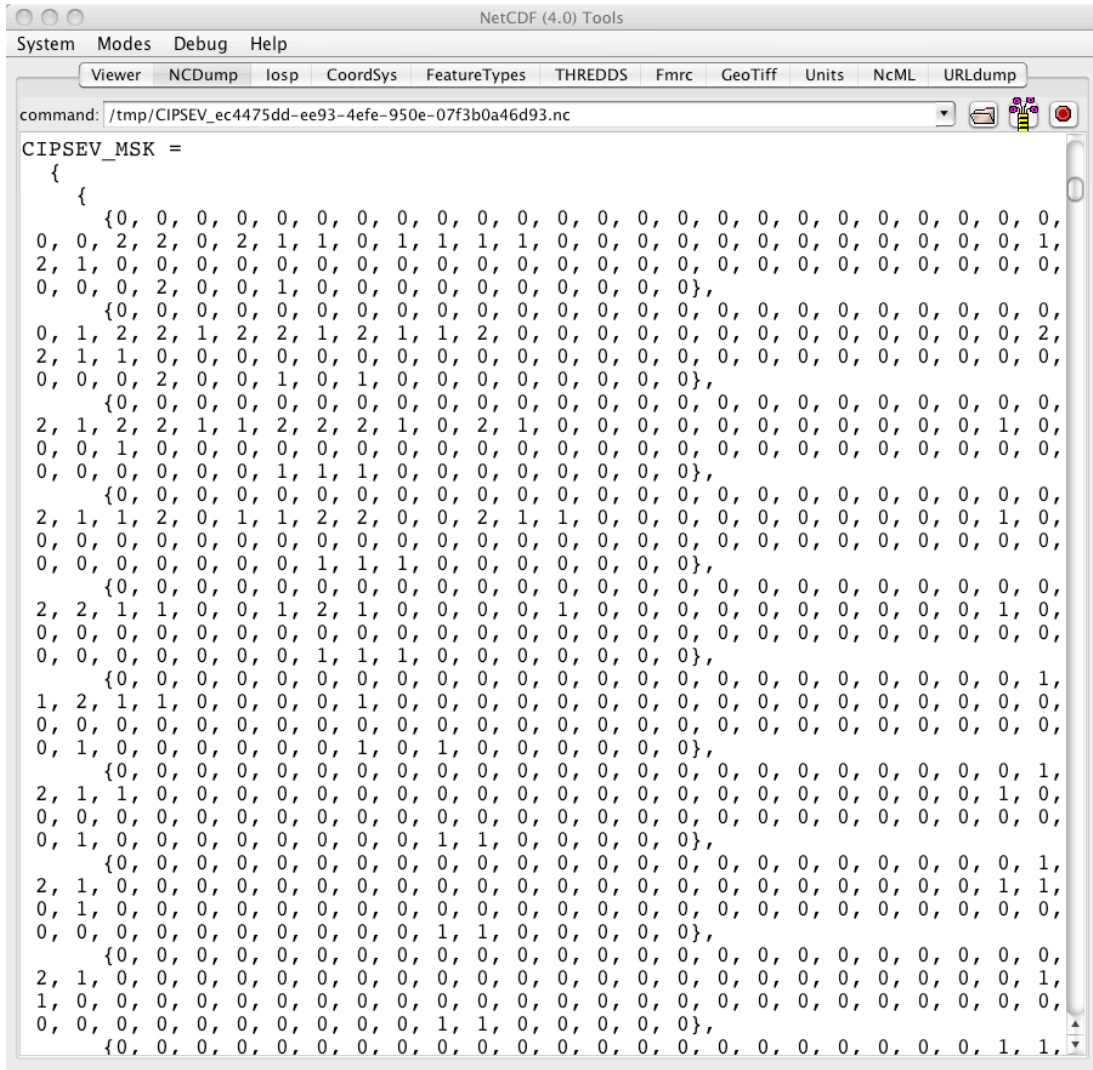


Figure 7.28 Thresholded Data

8. Ontology Alignment Tool

This section demonstrates the Ontology Alignment Tool.

8.1 Test Environment and Setup

8.1.1 Ontology Alignment UI

The Ontology Alignment UI can be downloaded from wxforge using SVN. The command to download the tool is:

```
svn checkout
http://wxforge.wx.ll.mit.edu/svn/ontologies/alignment\_tools/trunk/alignment\_tools
```

The remainder of this section will describe the installation as explained in the Ontology Alignment UI's installation.txt file.

Once the dependencies have been installed and the source downloaded, the UI can be installed. The “install” shell script in the UI's home directory uses maven to install dependencies in the local repository and compile the source. If not installed, type the following command from the \$UI_HOME directory (the directory to which the tool was downloaded):

```
./install
```

Then, to run the UI, type the following command also from the \$UI_HOME directory:

```
./ontology_gui
```

The installation can be tested with the NNEW weather ontology and the Climate and Forecast (CF) ontology by loading these two ontologies—found in the \$UI_HOME/src/main/resources directory—as described in section 7.1.2.

8.2 Loading Ontology

To begin using the UI, first load a source and target ontology. Begin by clicking on the "**Ontologies**" menu item in the upper left-hand corner of the UI. Select "**Load Ontology...**" and navigate to either the sample ontologies in the \$UI_HOME/src/main/resources directory, the ontologies downloaded from wxforge, or any other valid .owl file. Select the .owl file that you would like to load and click the "**Open**" button. Then, back on the main UI window, click the "**Open As Source**" button. Repeat the same process, this time selecting another ontology and clicking "**Open as Target.**" (Note: You can open the same ontology as both the source and the target if desired.)

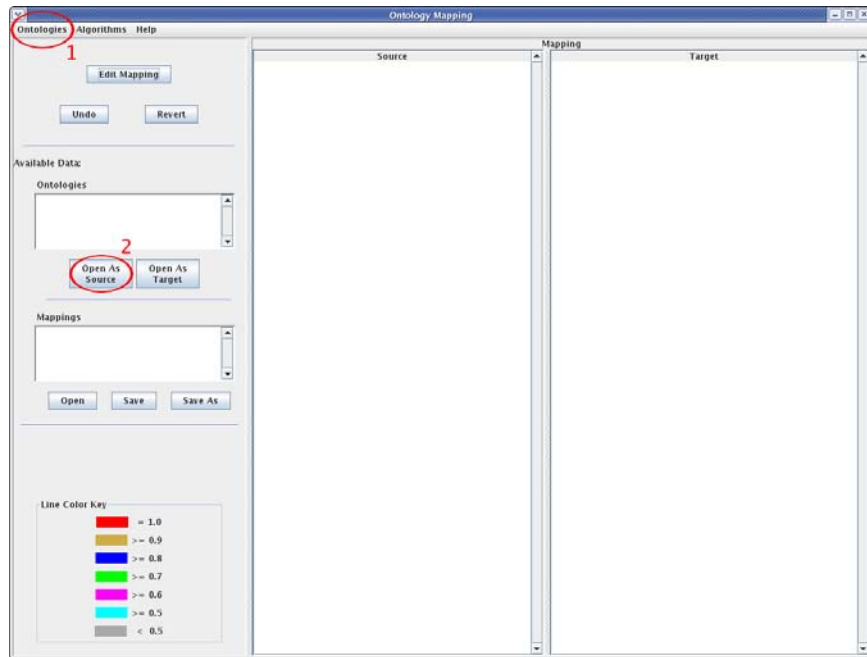


Figure 8.1 Ontology Alignment UI: Load ontology

At this point, two ontologies will be loaded side-by-side into the panes, the source on the left and the target on the right. You can expand the hierarchies in the tree to see more about the tuples each class has; the child node in the hierarchy shows the subclass restriction on the class.

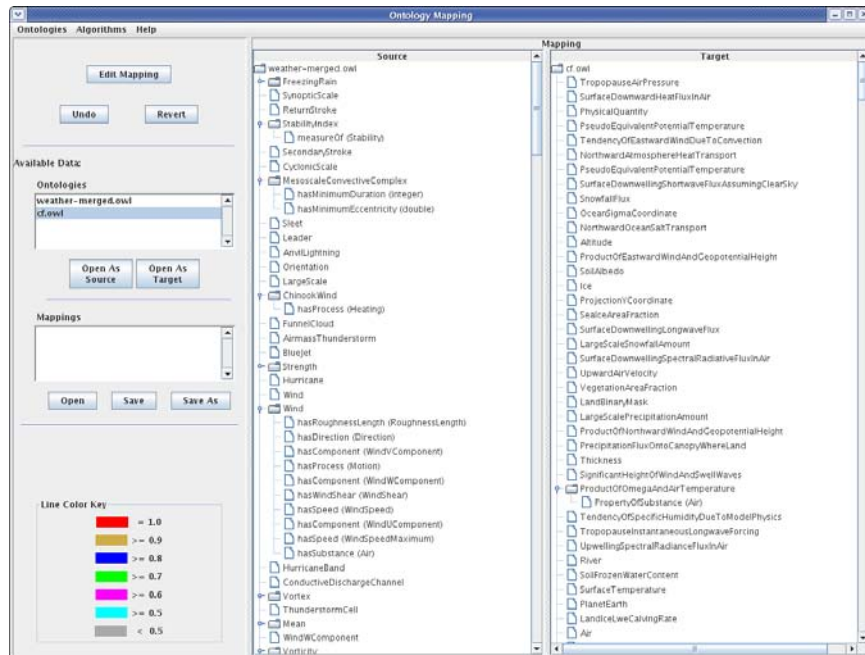


Figure 8.2 Ontology Alignment UI - Expand tuple hierarchies

From here, you can either use a semi-automatic alignment algorithm to detect matches between the ontologies or align them manually, which will be described in the following sections.

8.3 Manual Alignment of Ontologies

First click the “**Edit Mapping**” button to enter editing mode. (Note: In this mode, you will not longer be able to expand or contract the hierarchies in the tree.) To create a match, click and release on a term in either the source or the target ontology and click and release on its match in the other ontology. This will draw a line between the two. To change the weight—the measure of semantic relatedness between the two terms—right click on the line and selected “**Change weight...**” Enter a new weight between 1.0 and 0.0 in the dialogue box that appears. The line will then change color to reflect the updated weight.

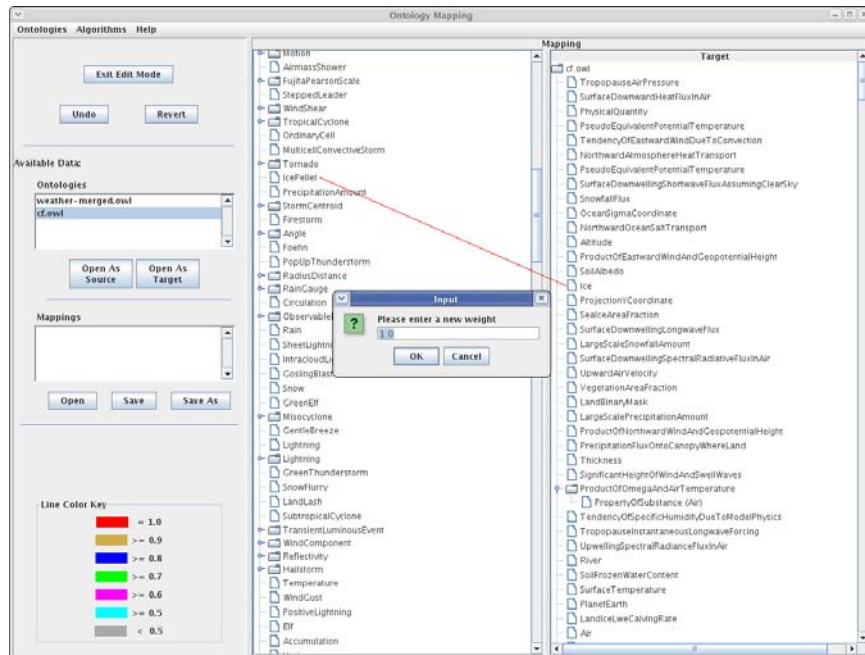


Figure 8.3 Ontology Alignment UI - Create matches and edit weights

Lines can be deleted by right-clicking on the line and selecting “**Remove Line.**”

After you have finished creating matches, you can save the alignment by clicking the “**Save As**” button under the “**Mappings**” section of the UI. Choose a name for the alignment file and click “**Save.**”

8.4 Semi-Automated Alignment of Ontologies

To run an algorithm on the two ontologies for semi-automatic alignment, navigate to the “**Algorithms**” menu. Select the algorithm from the menu that you would like to use. You can edit the configuration of the algorithm by clicking the “**Configure**” button. When you have finished the configuration, click “**Run.**”

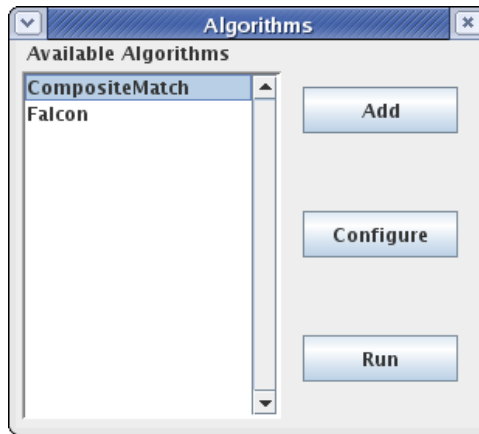


Figure 8.4 Ontology Alignment UI - Select alignment algorithm

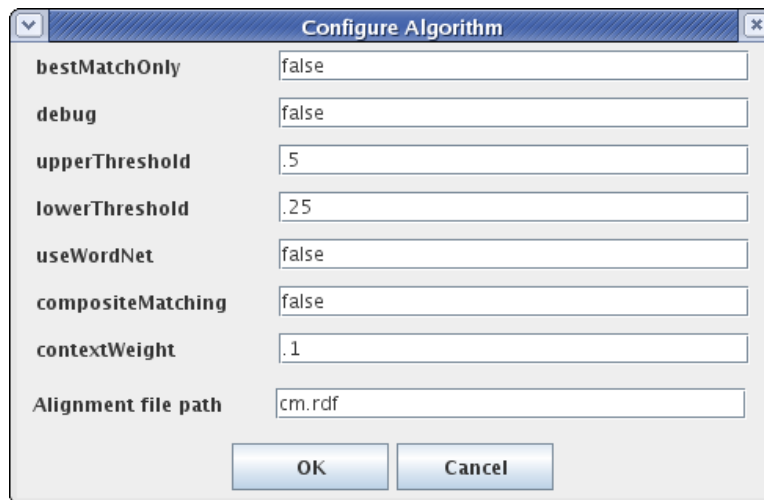


Figure 8.5 Ontology Alignment UI - Configure algorithm

After the algorithm finishes running, the results from the alignment algorithm will appear on the screen as matches—lines of varied weight between semantically similar concepts. These matches can be edited as described above using the manual edit mode. These alignments will be automatically saved to the filename you ascribe them the “**Alignment file path**” box in the configuration screen and will show up in the “**Mappings**” box on the main screen of the UI.

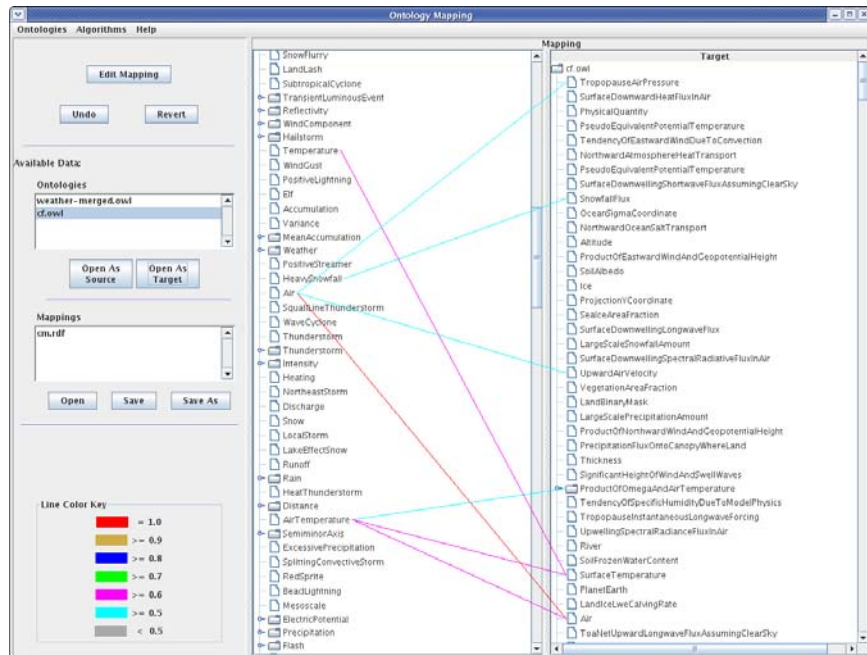


Figure 8.6 Ontology Alignment UI - Run alignment algorithm

8.5 Loading an Existing Alignment

Pre-existing or previously saved alignments can be loaded into the UI for viewing or editing. First, load the two ontologies to which the alignment pertains. Then, under the “**Ontologies**” menu, click “**Load Mapping...**” Navigate to the alignment file between the two ontologies, select it, and click “**Open.**” Then, click on the “**Open**” button under the “**Mappings**” section of the main UI screen. The matches will appear as lines between the two ontologies.

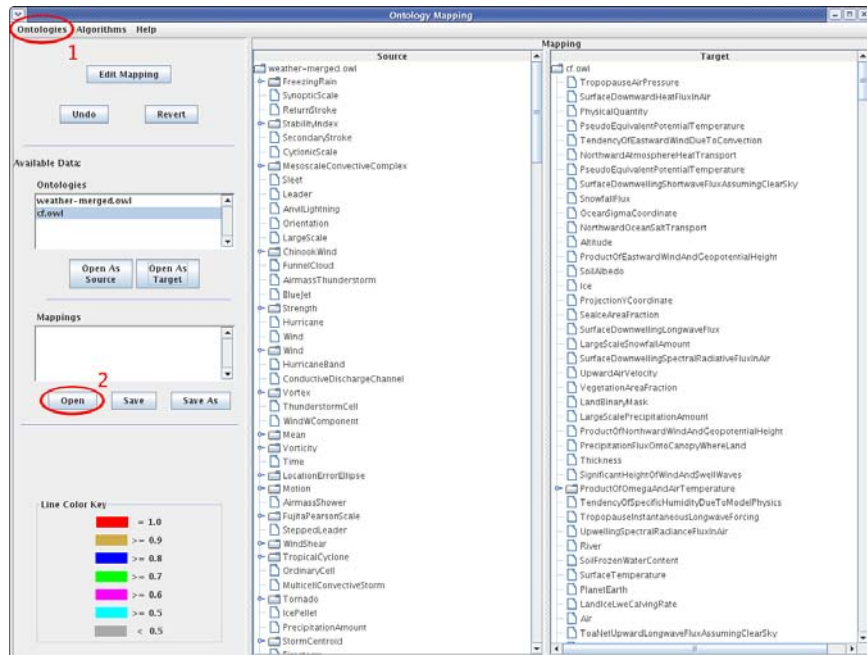


Figure 8.7 Ontology Alignment UI: Load mapping (Step 1)

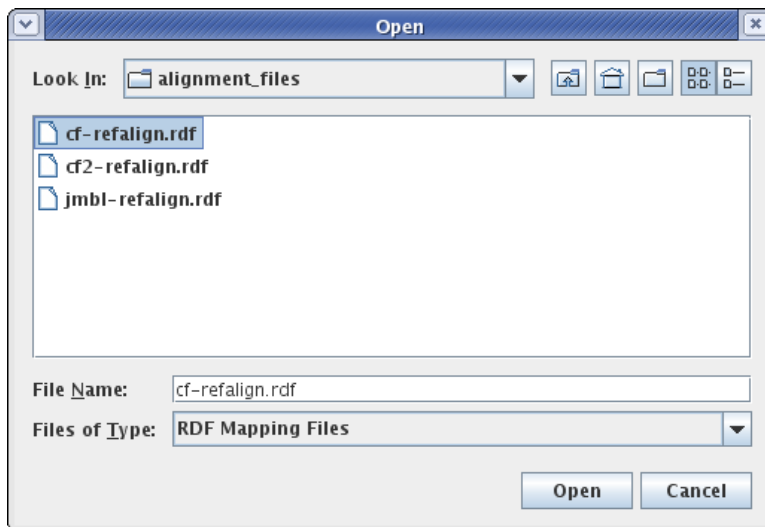


Figure 8.8 Ontology Alignment UI: Load mapping (Step 2)

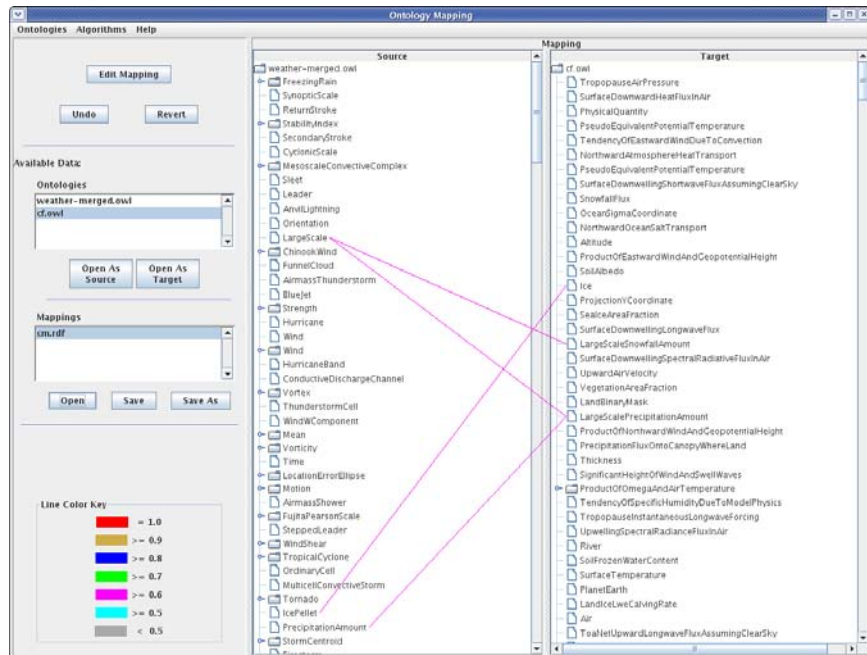


Figure 8.9 Ontology Alignment UI: Load mapping (Step 3)

9. References

JPDO Weather Functional Requirements Team. "Four Dimensional Weather Functional Requirements for NextGen Air Traffic Management" Joint Planning and Development Office. Jan 18, 2008

<http://www.jpdo.gov/newsArticle.asp?id=97>

NNEW Development Team. "NextGen Network-Enabled Weather Use Cases" May 20, 2008

<http://wiki.ucar.edu/download/attachments/17760853/NNEW-UseCases-v3.doc?version=3>