



Review of NASA ECHO (version 8) Web Service Registration Process

NASA Tech Infusion Working Group (TIWG) - Web Services Subgroup

Dr. Yuqi Bai and Dr. Liping Di
Center for Spatial Information Science and Systems
George Mason University

Ken Keiser
Information Technology and Systems Center
University of Alabama in Huntsville

July 2007

Index

Introduction.....	2
Background.....	3
Registration Authorization.....	3
Registration Preparation.....	4
Interactive Registration Process.....	6
Programming Interface Registration Process.....	8
Lessons Learned.....	10
Further discussion.....	11
Conclusions.....	13

Introduction

Web Service is a software system designed to support [interoperable Machine to Machine](#) interaction over a [network](#). It provides a promising plan to promote the online discovering and sharing of massive valuable geospatial data. The ability to register, discover, and govern these Web services is desirable to be in place with the number of available geospatial Web Services keeps growing. To promote the integration and use of Web Services for software applications within the same community, it is considered important to make service information easily available to researchers and developers.

The NASA Tech Infusion Working Group (TIWG) Web Services subgroup has undertaken the task of evaluating what might be the best approach or making web service information available to the Earth Science community. Availability of this information could be in the form of a simple listing, website or a type of feature-rich registry application. While easy to implement, simple listings of web services and corresponding documentation become difficult to manage and will ultimately not provide a standardized or consistent interface for future utilization.

Web Service registry, as a key infrastructural component and cornerstone for Service-Oriented-Architecture deployments, meets this critical need. It provides a distributed/centralized approach for the management of the metadata information of available Web Services. On the back end, it enables Web Service providers to register their distinct Web Services by providing descriptive information, also named metadata; on the front end, it provides a suite of public standards-compliant access interfaces for clients to discover this metadata information. The *Universal Description, Discovery and Integration* (UDDI) specification is one of the available registry interface standards, and NASA Earth Observation System (EOS) ClearingHouse (ECHO) is one of available Web Service registries.

The web services community has defined the UDDI specification to address the needs identified by the TIWG Web Services subgroup. While UDDI apparently has not yet realized wide acceptance or utilization, it appears to still be the most widely accepted technology for organizing service-related information. NASA's ECHO project has adopted UDDI for service directory functionality and has provided registration tools in support of users submitting services information. The TIWG Web Services subgroup tasked itself to investigate the service registration capabilities provided by ECHO. Subgroup participants from the University of Alabama in Huntsville (UAH) and George Mason University (GMU) conducted registration exercises with the ECHO interfaces and reported on those experiences. This is the report compiles the results from those investigations that were performed from December 2006 to January 2007.

Background

Initial access to the ECHO web site (<http://www.echo.nasa.gov>) provides an obvious starting point with a top level navigational link to “Service Partners”. This page provides a brief explanation of a service partner and explains an “ECHO Extended Service”. From this page there is a link to “Getting Started...” and other related pages.

ECHO (version 8) provides both programming interface and user interface approaches of registering web services. Both of these approaches are discussed in the sections below.

The ECHO Extended Service Management module allows ECHO service partners to introduce and maintain four Extended Service Business Objects: **Web Service interface**, **Web Service Implementation**, **advertisement** and **Web Service GUI**.

- A web service interface defines a web service API. It defines the messages and parameters necessary for using it. A web service interface is defined by a Web Service Definition Language (WSDL) 1.1 document that contains the types, import, message, portType, and binding elements. It is an abstract definition of a Web service. Web Service interfaces may be defined and maintained by ECHO partners, but are hosted and managed within the ECHO system.
- A web service implementation is a web service that is hosted by an ECHO partner. Like a web service interface, the service definition is also described by a WSDL document. The WSDL web service implementation document will contain one service element at a minimum.
- An advertisement is used to provide broadcasting or publications about data. The advertisement services are associated with the data items that they publicized; they also provide descriptions of the broadcasted services. Since advertisements are mainly description of services, they are not described or associated with WSDL files. They do have an access point which can be any string.
- A web service GUI is a user interface for Web Services that may be registered in ECHO.

Registration Authorization

ECHO has imposed limitations on who can register and provide information on services to the system. This is to preserve the integrity of the system and help manage the content.

To register Web Services in ECHO, two types of accounts are required. One is the Service Provider account, and the other is user account. These two must be associated together. A completed Service Provider Application document is emailed to ECHO Ops who then created a Service Provider account for testing purpose. A user account is created and associated with the Service Provider

account. Please refer to this page http://www.echo.eos.nasa.gov/services/service_start.shtml for more detailed information. A brief synopsis of the user registration steps are provided below:

1. Read and accept the Operations Agreement (OA)
<http://www.echo.nasa.gov/services/serviceDocs/ServiceOA1.pdf>
2. Fill out ECHO Service Partner Application form
<http://www.echo.nasa.gov/services/serviceDocs/ServicePartnerApplication.pdf>
3. Email application to echo@killians.gsfc.nasa.gov
4. Receive response saying the application was complete and would be reviewed.
5. Receive confirmation of application approval ECHO Ops.

Registration Preparation

Existing services and the corresponding definitions from both GMU and UAH were identified as candidates for the ECHO registration efforts. GMU provided an *ImageOverlay* Web Service which was developed and used at the Center for Spatial Information Science and Systems (CSISS), GMU. This service is capable of overlaying two or more images that may have different lat/long bounding boxes and/or different transparent colors. The input images can be of JPEG, GIF, BMP or PNG formats; and the output images are always PNG images. The WSDL file of this Web Service is at: <http://laits.gmu.edu:8099/axis/ImageOverlay.iws?wsdl>. UAH focused on registering subsetting and data mining services. The subsetting tools are developed specifically to handle HDF-EOS data sets while the data mining tools are not data format specific. The WSDL for the subsetting tools is located at <http://ws.itsc.uah.edu/services/subsetting/HEW/HEWSubsetting.wsdl> and multiple WSDL for data mining tools exists at <http://ws.itsc.uah.edu:3945/services/mws>.

Usually, the Web Service Interface and the Web Service Implementation definitions are in the same WSDL file. To be compatible with ECHO it was necessary to re-organize the existing WSDL to get two separate definitions for the interface and implementation sections. Below is an example of this step using the GMU *ImageOverlay* Service WSDL.

The content in the blue rectangle (Figure 1) is the Web Service Implementation section of the *ImageOverlay* Service WSDL. The others are the Web Service Interface definitions.

```

<?xml version="1.0" encoding="UTF-8" ?>
- <wsdl:definitions targetNamespace="http://laits.gmu.edu:8099/axis/ImageOverlay.jws"
  xmlns:apachesoap="http://xml.apache.org/xml-soap"
  xmlns:impl="http://laits.gmu.edu:8099/axis/ImageOverlay.jws"
  xmlns:intf="http://laits.gmu.edu:8099/axis/ImageOverlay.jws"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:wSDL="http://schemas.xmlsoap.org/wsdl/"
  xmlns:wSDLsoap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
+ <!-- -->
+ <wsdl:types>
+ <wsdl:message name="OverlayMultipleImagesWithSameRegionRequest">
+ <wsdl:message name="OverlayMultipleImagesRequest">
+ <wsdl:message name="OverlayTwoImagesResponse">
+ <wsdl:message name="OverlayMultipleImagesResponse">
+ <wsdl:message name="OverlayMultipleImagesWithSameRegionResponse">
+ <wsdl:message name="OverlayTwoImagesRequest">
+ <wsdl:portType name="ImageOverlay">
+ <wsdl:binding name="ImageOverlaySoapBinding" type="impl:ImageOverlay">
- <wsdl:service name="ImageOverlayService">
- <wsdl:port binding="impl:ImageOverlaySoapBinding" name="ImageOverlay">
  <wsdlsoap:address location="http://laits.gmu.edu:8099/axis/ImageOverlay.jws" />
  </wsdl:port>
</wsdl:service>
</wsdl:definitions>

```

Figure 1 ImageOverlay WSDL

The interface definition is produced by simply removing the implementation section. The result is shown in figure 2. We named this file as *ImageOverlay-interface.wsdl*.

```

<?xml version="1.0" encoding="UTF-8" ?>
- <wsdl:definitions targetNamespace="http://laits.gmu.edu:8099/axis/ImageOverlay.jws"
  xmlns:apachesoap="http://xml.apache.org/xml-soap"
  xmlns:impl="http://laits.gmu.edu:8099/axis/ImageOverlay.jws"
  xmlns:intf="http://laits.gmu.edu:8099/axis/ImageOverlay.jws"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:wSDL="http://schemas.xmlsoap.org/wsdl/"
  xmlns:wSDLsoap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
+ <!-- -->
+ <wsdl:types>
+ <wsdl:message name="OverlayMultipleImagesWithSameRegionRequest">
+ <wsdl:message name="OverlayMultipleImagesRequest">
+ <wsdl:message name="OverlayTwoImagesResponse">
+ <wsdl:message name="OverlayMultipleImagesResponse">
+ <wsdl:message name="OverlayMultipleImagesWithSameRegionResponse">
+ <wsdl:message name="OverlayTwoImagesRequest">
+ <wsdl:portType name="ImageOverlay">
+ <wsdl:binding name="ImageOverlaySoapBinding" type="impl:ImageOverlay">
</wsdl:definitions>

```

Figure 2 ImageOverlay Interface WSDL

The service implementation definition was generated by keeping the root element (*wsdl:definitions*) and the service element (*wsdl:service*), removing all the other elements from the *ImageOverlay* WSDL, and added a new *import* element. The result shown in Figure 3 is the *ImageOverlay-implementation.wsdl*.

```

<?xml version="1.0" encoding="UTF-8" ?>
<wsdl:definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:apachesoap="http://xml.apache.org/xml-soap"
  xmlns:impl="http://laits.gmu.edu:8099/axis/ImageOverlay.jws"
  xmlns:intf="http://laits.gmu.edu:8099/axis/ImageOverlay.jws"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:wSDL="http://schemas.xmlsoap.org/wsdl/"
  xmlns:wSDLsoap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://laits.gmu.edu:8099/axis/ImageOverlay.jws">
  <import namespace="http://laits.gmu.edu:8099/axis/ImageOverlay.jws"
    location="http://laits.gmu.edu/~ybai/echo8ws/ImageOverlay-interface.wsdl"/>
  <wsdl:service name="ImageOverlayService">
    <wsdl:port name="ImageOverlay" binding="impl:ImageOverlaySoapBinding">
      <wsdlsoap:address location="http://laits.gmu.edu:8099/axis/ImageOverlay.jws"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>

```

Figure 3 ImageOverlay Implementation WSDL

The `namespace` attribute value of `import` element is a URL that matches the `targetNamespace` attribute of the root element (`wsdl:definitions`) in the ImageOverlay Interface WSDL (ImageOverlay-interface.wsdl).

Interactive Registration Process

The ECHO web site provides a downloadable workstation-based utility, the *Extended Services Tool for ECHO 8.0*. The tool allows service providers to fulfill service registration through a form-based interface. Although the website describes the application as a “web-based” tool, this ECHO registration tool is an actually desktop program. *No mention of operating system* is given on the website, but users will discover it runs within Microsoft .Net Framework environment and interacts across the network with the ECHO system. It is downloadable from ECHO website¹. Microsoft .Net Framework environment can be downloaded from Microsoft website².

After uncompressing the zip package, you will find the executable file: `EchoClient.exe`. Run this file and it will bring a window titled “ECHO ES Client V1.2.2499.24993”, as shown in Figure 4. At last attempt, the default “ECHO URL” provided on the application’s startup form is incorrect and a correct address must be obtained from the ECHO Ops team.

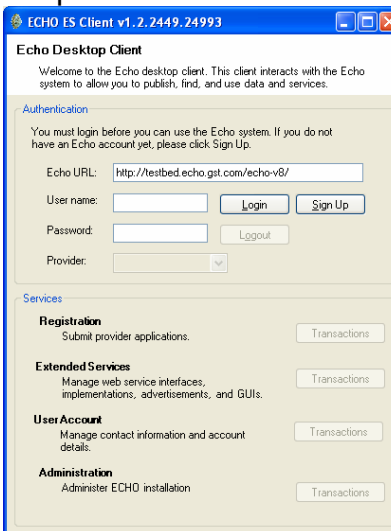


Figure 4 ECHO Extended Services Tool (a)

Web Service Interface Registration

The user provides the (correct) “Echo URL” value and the “User name” and “Password”, and then presses the Login button. The client application then logs into the ECHO system, and populates the Provider list. The user selects the appropriate “Provider”, as shown in Figure 5.

¹ http://www.echo.nasa.gov/services/service_tools.shtml

² <http://www.microsoft.com/downloads/details.aspx?FamilyID=0856each-4362-4b0d-8edd-aab15c5e04f5&displaylang=en>

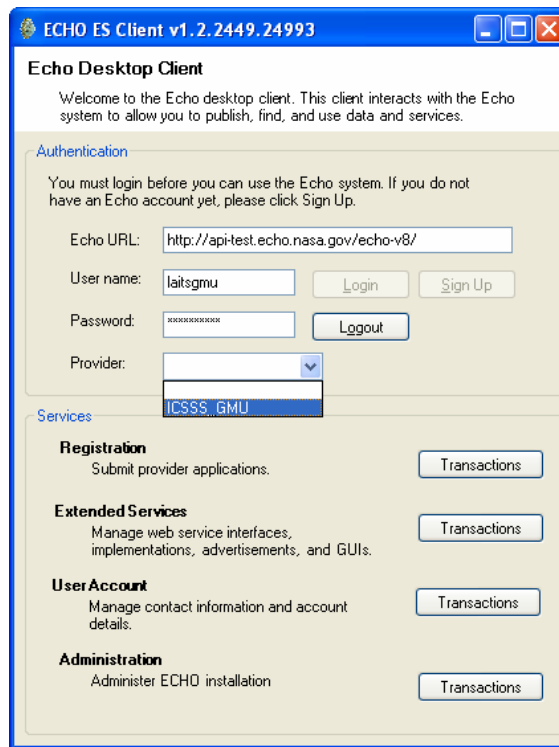


Figure 5 ECHO Extended Services Tool (b)

Click on each Transactions button will bring a context menu. The menu for ExtendedService Service is shown in Figure 6:

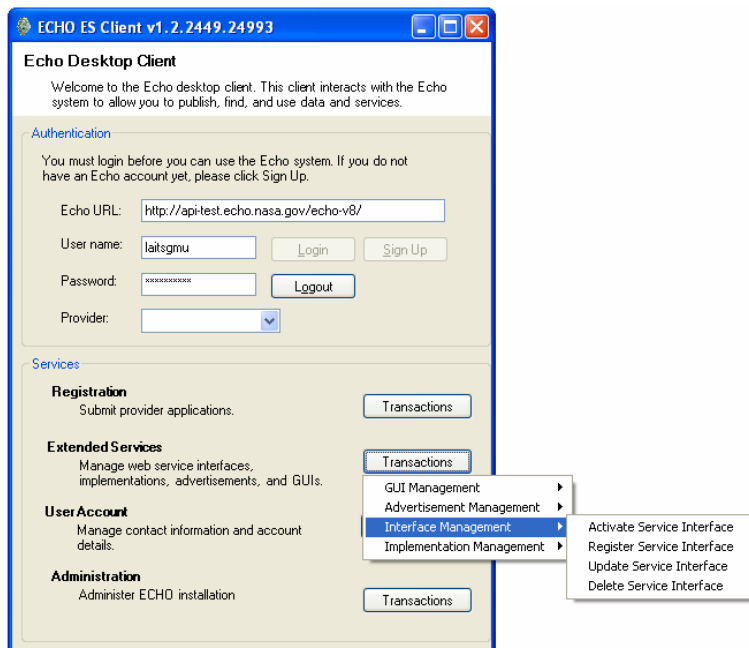


Figure 6 Extended Service Management Menu in ECHO Extended Services Tool

Clicking each content menu entry will bring a wizard. Taking “Interface Management -> Register Service Interface” for example, it brings a new window titled Register Web Service Interface, as shown in the figure below:

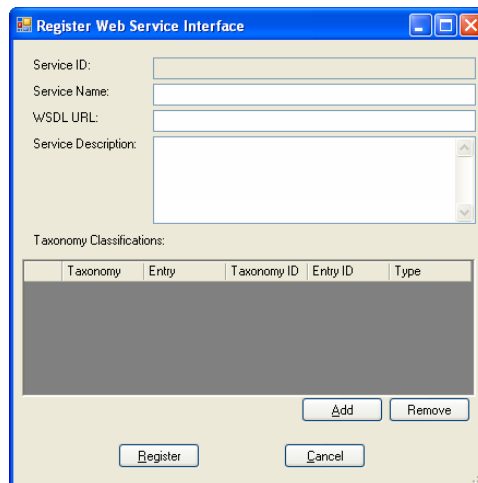


Figure 7 Interface Registration Wizard in ECHO 8 through ECHO Extended Services Tool

Following each wizard, Web Service Provider can fulfill register/update/delete tasks for four Web Service Objects.

Programming Interface Registration Process

ECHO provides a programming interface that can be utilized to programmatically register services with the ECHO system. The GMU team programmed an example using this approach and provided the following description.

The GMU team developed Java programs to register their *ImageOverlay* Web Service Interface and the Implementation into the ECHO 8 Partner Test System, with the support from the Apache Axis library [<http://ws.apache.org/axis/>]. They used WSDL2Java tool to generate all the Java client bindings from the ECHO 8 WSDL files. All these bindings were compiled and packed as a Java Jar library. This Jar library was used by the Java programs to communicate with ECHO 8 ExtendedService Service and other Services such as Authentication Service. All the ECHO 8 Services WSDL files can be found at: <http://www.echo.eos.nasa.gov/reference/reference.shtml>.

1. Web Service Interface Registration

The Web Service Interface Registration operation API interface looks like:

[string CreateWSInterface \(string token, string wsdlURL, string serviceName, string serviceDescription, ListOfClassifications classifications\)](#)

Here, *wsdlURL* is the URL of the Web Service Interface WSDL file. *serviceName* and *serviceDescription* are used to identify this interface. *classifications* parameter is an array of the Classification object.

In ECHO 8, **Classification** is closely related to the **Taxonomy Entry** and **Taxonomy**. Each taxonomy has a tree of taxonomy entries. A taxonomy entry consists of a name and a value. The name of the taxonomy entry is the human readable name. The value is what makes the taxonomy entry unique in a

taxonomy. A taxonomy entry name can be repeated in the same taxonomy but the value cannot. A classification is just the name given when classifying an extended service entity with a taxonomy entry.

There are three taxonomies in ECHO, as shown in table 1.

Table 1 ECHO 8 Taxonomies

Taxonomy Name	Description
nasa-ECHO:Dataset	This is the primary input type dataset taxonomy. This should be used to classify services where the dataset is the primary input to the service.
nasa-ECHO:ServiceType	This is a categorization scheme of kinds of services from a functional perspective. It is comprised of other taxonomies defined by external organizations.
nasa-ECHO:DataFormat	This is a categorization scheme based on the format of Earth Science data. It is similar to a mime-type classification scheme.

If classification info is not provided when registering Web Service Objects, it can be added later through update operation. The response from the [createWSInterface](#) operation is a string, named GUID. It looks like this: [1AC3CDC1-C4C1-C851-5445-A19F7323F477](#). It uniquely identifies this Web Service interface in ECHO. The service provide needs to keep this GUID for the future maintenance use (discovery, update, delete).

2. Web Service Interface Discovery (I)

To validate the success of the registration, we invoked the discovery operation, [getWSInterfaces](#), with that interface GUID as an input parameter. However an “[Interface 1AC3CDC1-C4C1-C851-5445-A19F7323F477 is not active](#)” Exception resulted from testing. It means that the Web Service Interfaces need to be activated after registration.

3. Web Service Interface Activation

To activate this *ImageOverlay* Web Service Interface, the team tried to invoke the [activateWSInterfaces](#) operation with that interface GUID as an input parameter. This attempt was met with another Exception as “[Must be an admin to active a web service advertisement](#)”. [Note: ECHO Ops have confirmed that, after each Web Service Object has been registered, they should be automatically notified. They are examining this issue. Hopefully this could be resolved soon. That is to say, Service Providers do not need to activate Web Service Objects by themselves. These objects will be automatically activated by ECHO Ops.]

4. Web Service Interface Discovery (II)

After being activated, this *ImageOverlay* Interface is public and discoverable. The example invoked the [getWSInterfaces](#) operation again and got one Web Service Interface returned as below:

[guid=1AC3CDC1-C4C1-C851-5445-A19F7323F477](#)

ProviderGuid=ICSSS_GMU
Name=GMU ImageOverlay Service
WsdUrl=http://api-test.echo.nasa.gov:80/echo-es-wsd/1AC3CDC1-C4C1-C851-5445-A19F7323F477
Description=This ImageOverlay Service can be used to overlay multiple images that may have different transparent colors, bounding boxes.

Please note that the **WsdUrl** has been changed. It is hosted by the ECHO system. Users need to reference this new Interface URL in the Implementation WSDL.

5. Modifying ImageOverlay Implementation WSDL

The **location** attribute value of the **import** element needed to be set to this new Interface URL. The new Implementation WSDL looks like this:

```
<?xml version="1.0" encoding="UTF-8" ?>
- <wsdl:definitions targetNamespace="http://laits.gmu.edu:8099/axis/ImageOverlay.jws"
  xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:apacheSOAP="http://xml.apache.org/xml-
  soap" xmlns:impl="http://laits.gmu.edu:8099/axis/ImageOverlay.jws"
  xmlns:intf="http://laits.gmu.edu:8099/axis/ImageOverlay.jws"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <import namespace="http://laits.gmu.edu:8099/axis/ImageOverlay.jws"
    location="http://api-test.echo.nasa.gov/echo-es-wsd/1AC3CDC1-C4C1-C851-5445-
    A19F7323F477" />
- <wsdl:service name="ImageOverlayService">
- <wsdl:port binding="impl:ImageOverlaySoapBinding" name="ImageOverlay">
  <wsdlsoap:address location="http://laits.gmu.edu:8099/axis/ImageOverlay.jws" />
  </wsdl:port>
</wsdl:service>
</wsdl:definitions>
```

Figure 8 New ImageOverlay Implementation WSDL

6. Web Service Implementation Registration

The *createWSImplementation* function is used to register this ImageOverlay Web Service Implementation. The response string, *94BC50BA-61BA-64BE-78ED-A663978F4CBC*, is a GUID string identifying this WS implementation in ECHO.

7. Web Service Implementation Discovery

After being activated, this *ImageOverlay* Web Service Implementation is public and discoverable. The *getWSImplementation* operation is invoked with that Web Service Implementation GUID as input parameter and got one Web Service Implementation returned.

Lessons Learned

Based on this exercise, our understanding is:

1. ECHO 8 supports Web Service Providers to register the Web Service Interface and the Web Service Implementation objects.
2. Web Service Providers need to have two types of accounts in ECHO: one Service Provider account and at least one user account. The user account(s) must be associated with the Service Provider account.
3. Two distinct WSDL files are needed: the Web Service Interface WSDL and the Web Service Implementation WSDL. They should be accessible on the web.

-
4. Web Service Interfaces will be activated by ECHO Ops. A new hosting URL will be provided by the ECHO system. We need then to update Web Service Implementation WSDL file accordingly to reference this new Web Service Interface URL.
 5. The Web Service Interfaces and the Web Service Implementations will not be publicly discoverable and accessible until they are activated by ECHO Ops.

Suggestions for Web Service Providers:

1. First register the Web Service Interface then the Web Service Implementation.
2. Run discovery function to guarantee that the registered Web Service Object is available before taking the next step.
3. Use ECHO Extended Services Tool to fulfill Web Service Object registration.
4. Send emails directly to ECHO Ops at echo@killians.gsfc.nasa.gov for any problems you may encounter when registering Web Service Objects.

Suggestions for ECHO Operations Group:

1. The ECHO Extended Services Tool only provides Register/update/delete functionalities for Web Service Providers. It would be better to support discovery functions.
2. Remove unnecessary functions from the ECHO Extended Services tool, or make them invisible to Web Service Providers.
3. Our experiment is time-consuming. We encountered some exceptions or problems in almost every step. A user guide for Web Service Providers is highly needed. It not only shows one step after another how they can fulfill all the tasks, but includes some QoA sections for quick problem resolving.
4. A guide for Web Service Users is also needed.
5. ECHO Ops personnel are very helpful. It would be better if they could be more responsive. Currently, users are supposed to report any issue to an open email list. But some times, "everybody is responsible" can come to mean "nobody is responsible".
6. The web site needs to correctly describe the "Extended Services Tool" as application based.
7. The Extended Services Tool provided on a specific ECHO version's web site needs to be modified to be in synch with that version in terms of defaults and capabilities.

Further discussion

1. Web Service Discovery Problem

Taking the Web Service Interface for example, there are only two operations supporting Web Service Interface discovery:

GetServiceNamesByTaxonomyEntry

GetWSInterfaces

The first one lists all of the Web Service objects (including Interfaces, Implementations, Advertisements, and GUIs) that are classified by the given

taxonomy entry. The second one retrieves the interfaces that match given interface GUIDs. For Web Service Users, only the first operation is applicable. This is because Interface GUIDs are only known to Web Service Providers, other than Web Service users.

Only support discovery through Taxonomy is not enough. Think about this question: how we can discover, evaluate and choose one OGC Web Coverage Service (WCS) from hundreds of or even thousands of registered WCS services that are all classified as OGC WCS? Returning all these services back to users is not a good design. This question reveals three issues: we need more meta-data information, metrics information and more discovery operations.

✧ Extra meta-data information

Each WCS service provides *coverage(s)* which cover either global area or a specific region. Such kinds of spatial coverage information would be very helpful when Web Service users discover OGC WCS implementations. If this information is not available in ECHO, Web Service Users have to query each WCS implementation manually to know whether or not their data requirement can be fulfilled.

✧ Extra metrics information

The quality of the Web Service is another issue when Web Service users evaluate each Web Service implementations. Those always-online, always-available Web Service Implementations should be honored and their Service Providers should get credited. All of these quality related information are very valuable for Web Service users.

One way to collect these metrics information is through [InvocationService](#) operation. This service can be enforced to provide feedbacks after each Web Service invocation.

✧ More discovery operations

There should be more choices to help Web Service users to find and to evaluate those registered Web Service Implementations. More query criteria should be supported, such as supporting the Service Discovery based on the Service Provider.

2. OGC Service Taxonomy

We suggest adding Version entries for the OGC Service taxonomy. That is because there may be great difference between different versions of the same OGC Implementation Specification.

3. “Service Type”

ECHO “Service Type” taxonomy only includes some well known Services that provide different functionalities. The service input parameters and out parameters are indirectly defined by referencing the well known Services specifications, e.g. OGC Web Coverage Service. From this point of view, these kinds of “Service Type” only apply for those agreed-upon Services. However, GMU has introduced a more detailed “Service Type” definition for each individual Web Service in **GeoBrain** project, no matter whether or not it is compliant with some well known

Service Specifications (NASA REASoN program, Principal Investigator: Professor Liping Di). This “Service Type” is a combination of the service input parameters and the service output parameters. This design is useful at fulfilling “**Virtual Data**” scenario, and has greatly facilitated semiautomatic and automatic Service Chaining. This design has been validated and realized in the **GeoBrain** production System.

Conclusions

The web experience for “new” service partners seems to be a little sparse on information, especially for someone not intending to write an application to perform the service registration. From additional information obtained later, there may be a disconnect and confusion about whether a user is working in the “production” system which is action version 9 of ECHO or the “testbed” system which is version 8. There appears to be virtually no indication of what system a user is working in nor an explanation of why you would want to be in one or the other, on the web site. The support for web service partners appears to still be in the introductory stage, so it is expected that the ECHO development group will welcome this and all feedback on the usability of their system and will aid them in addressing issues for different communities and user groups. Expansion of the available taxonomy is crucial to be able to properly classify other services, for instance the data mining (analysis) services are not specialized for any “data set” or “data format” and they do not fall into the available “service types” (see Table 1). Individual requests to have the taxonomy definition expanded have not been addressed so this may need to be a committee-based action to see results.