



## Client Hands On Lab









## Agenda

Thursday, November 6 <sup>th</sup> , 2003 – Client Hands On Lab					
11/6/2003	9:00 AM	1:00	10:00 AM	<a href="#">User Accounts</a>	
11/6/2003	10:00 AM	0:30	10:30 AM	<a href="#">Catalog Service</a>	
11/6/2003	10:30 AM	0:15	10:45 AM	Break	
11/6/2003	10:45 AM	1:00	11:45 AM	More Catalog Service	
11/6/2003	11:45 AM	1:30	1:15 AM	Lunch	
11/6/2003	1:15 PM	0:30	1:45 PM	<a href="#">Provider Profile Service</a>	
11/6/2003	1:45 PM	1:00	2:45 PM	<a href="#">Placing Orders</a>	
11/6/2003	2:45 PM	0:15	3:00 PM	Break	
11/6/2003	3:00 PM	0:45	3:45 PM	More Orders	
11/6/2003	3:45 PM	1:00	4:45 PM	<a href="#">Subscriptions</a>	


Global Science & Technology, Inc.




## User Account Service









## Why do we have the User Account Service?

- The User Account Service is used to manage attributes associated with users
  - Addresses
  - Telephone numbers
  - Some aspects of orders
  - Preferences
  - Passwords


Global Science & Technology, Inc.




## Example #1: Managing Addresses

- Login as DemoUser#
- Let's look at the entire user
- This does not include the Addresses or Telephone numbers associated with the user.
- File: 01\_PresentUserInfo.xml


Global Science & Technology, Inc.




## Presenting Your Addresses

- You can either present all the addresses associated with your account, or a specific address.
- For now, let's present all addresses.
  - You will find that your home address has already been filled in for you
  - Your home address was specified in the CreateUser message used to create the account
- File: 02\_PresentAddressInformation.xml


Global Science & Technology, Inc.


## Adding an Address

- A user can have multiple addresses associated with their account.
- Let's add some a work address
- File: 03\_AddAddress.xml



## Verify the new Address

- To double-check the 'work' address, issue the PresentAddressInformation transaction
- File: 04\_PresentAddressInformation.xml



## What if entered the wrong address?

- Issue the UpdateAddressInformation transaction to correct the wrong address.
  - Completely overwrites the previous address
- File: 05\_UpdateAddressInformation.xml



## Verify the changed Address

- Re-Issue the PresentAddressInformation transaction to verify the change occurred
- File: 06\_PresentAddressInformation.xml



## Clean-Up

- Delete the Address
- File: 07\_DeleteAddress.xml



## Example #2: Managing Telephone Numbers

- Follows the same pattern as managing your address information
- Let's start by displaying our current telephone information
  - You will find that you have a phone number already entered for 'home'
- File: 01\_PresentPhoneInformation.xml



## Adding a Telephone Number

- Users may have multiple telephone numbers for their wireless phone, home phone, and work phone
- Let's add a work phone to our profile
- File: 02\_AddPhone.xml

## Verify the new Phone

- Re-Issue the PresentPhoneInformation transaction to verify the work phone was created
- File: 03\_PresentPhoneInformation.xml

## Updating a Telephone Number

- There are two cases where a telephone number will need to be changed
  - The user entered it incorrectly
  - The user has changed their telephone number
- To change a telephone number, use the UpdatePhoneInformation transaction
- File: 04\_UpdatePhone.xml

## Verify the changed Phone

- Re-Issue the PresentPhoneInformation transaction to verify the work phone was changed
- File: 05\_PresentPhoneInformation.xml

## Clean-Up

- Delete the 'work' phone
- File: 06\_DeletePhone.xml

## The Rest of the User's Information

- We still haven't covered the basic user information attributes
  - First Name
  - Last Name
  - Email Address
  - Password

### Example #3: Changing User Information

- **Present the User's Information.**
- **You will find:**
  - First Name: User
  - Last Name: One
  - Email Address: [echodeveloper1@yahoo.com](mailto:echodeveloper1@yahoo.com)
- **File: 01\_PresentUserInformation.xml**



### Change the User's Information

- **Set it to:**
  - First Name: NewUser
  - Last Name: NewOne
  - Email Address:
    - [someotheruser@someotherhost.com](mailto:someotheruser@someotherhost.com)
- **File: 02\_UpdateUserInformation.xml**



### Verify the Change

- **Present the User's Information**
- **You will find:**
  - First Name: NewUser
  - Last Name: NewOne
  - Email Address:
    - [someotheruser@someotherhost.com](mailto:someotheruser@someotherhost.com)
- **File: 03\_PresentUserInformation.xml**



### What about Passwords?

- They are changed by the ChangePassword transaction
- You can verify the change by logging out and logging back in with the password 'welcome1'
- **File: 04\_ChangePassword.xml**



### Clean-Up

- **Revert the User Information**
- **Revert the password**
- **File: 05\_UpdateUserInformation.xml**
- **File: 06\_ChangePassword.xml**



### Orders

- **Most management of orders is through the Order Entry Service.**
- **Some transactions exist in the User Account Service to display pending order history and pending orders**



## Example #4: Working with Orders

- **Previously Created Orders**
  - PresentOrderHistory
- **Orders that are still 'open'**
  - PresentPendingOrderHistory
- **To get started, let's create an order**
  - Make sure you capture your OrderId to your Worksheet
- **File: 01\_CreateOrder.xml**



## Present Your Order History

- **Invoke the PresentOrderHistorySummary transaction**
- **All your closed orders are presented**
  - You will note that your recently created order does NOT appear in the list (because it is not closed)
- **File: 02\_PresentOrderHistorySummary.xml**



## Submit Your Order

- **SetUserInformation, Validate, and Submit your order**
  - Make sure you paste your OrderId into the messages
- **The order should immediately be closed**
- **File: 03\_SetUserInformation.xml**
- **File: 04\_ValidateOrder.xml**
- **File: 05\_SubmitOrder.xml**



## Present Your Order History

- **Invoke the PresentOrderHistorySummary transaction**
  - The order you just submitted should show up in the list
- **File: 06\_PresentOrderHistorySummary.xml**



## What about Broken Orders?

- **To display orders that contain exceptions, use the PresentOrderHistorySummary**
- **Explicitly list the TerminatingStates**
  - CLOSED
  - CLOSED\_WITH\_EXCEPTIONS
- **File: 07\_PresentOrderHistorySummary.xml**



## Example #5: Pending Orders

- **Very Similar to Order History Summary**
- **Shows all orders that are not yet closed**
- **Let's present any pending orders that you may have in your account**
- **File: 01\_PresentPendingOrders.xml**



## Create an Order

- **SetUserInfo, Validate, and Submit your order**
  - Make sure you paste your OrderId into the messages
- **Unlike before, your order has not been closed because it was made against a provider that does not immediately close orders**
- **File: 02\_CreateOrder.xml**
- **File: 03\_SetUserInfo.xml**
- **File: 04\_ValidateOrder.xml**
- **File: 05\_SubmitOrder.xml**



## Present Pending Orders

- **Like Order History, you can present all your pending orders, or pending orders that exist in specific states**
- **Let's present all of our pending orders**
- **File: 06\_PresentPendingOrders.xml**



## Example #6: User Preferences

- **This is an undocumented, untested feature**
- **Our generic option framework allows ECHO to define a structure which users can fill in with values**
- **The option framework is a major component of the order entry service subsystem and the provider policy subsystem**
- **For the purposes of this demonstration we have whipped something together for your delight**
  - We created some option definitions for users
    - Default shipping, billing, and contact address



## Example #7, User's Roles

- **ListRoles: list all of the roles user has, Currently users can have either Provider Roles or Admin Roles.**
- **File: 01\_ListRoles.xml**



## Option Hierarchy

- **We create the option definitions**
  - In this case, there are 3 definitions we defined, default\_shipping\_address, default\_billing\_address, default\_contact\_address
- **You can then make an option selection, which fills in values for the abstract option**
  - You can put the AddressID of user's addresses into the selection
- **The option selection can then be generically referenced later on**
  - This could be a significant value-add for clients



## Present the Option Definitions

- **To present the structure of the addresses, use the PresentOptionDefinitionsForUser transaction**
  - It follows the same pattern as other ECHO transactions wherein failure to specify a specific option definition presents all option definitions
- **File: 01\_PresentOptionDefinitions.xml**



## Specify Preferences

- **Let's go ahead and tell ECHO about some default addresses**
  - In reality, you would program your client to read a particular preference definition that it has knowledge of
  - At run time, your client would fill in the preferences as per user request
- **Create a billing, contact, and shipping preferences**
- **File: 02\_SetOptionSelections.xml**



## Present Your Selections

- **Issue the PresentOptionSelectionsForUser transaction to display what preferences the user has.**
  - In reality, you would program your client to read those preferences which your client understands
  - The values read could then be used as input to further transactions
- **File: 03\_PresentOptionSelections.xml**



## What? I don't understand...

- **What can you (as a client developer) use preferences for?**
  - Imagine a client that displays the United States
  - When a customer uses your system for the first time, your client forces them to make tell you what state they hail from
  - You could determine a bounding box from the state they chose, and store that information in ECHO as a default\_spatial\_region preference
  - In subsequent queries to the system, your client can automatically append the default\_spatial\_region preference you retrieved from ECHO
    - The alternative is that you (as a client provider) would have to persist the extra information through some other mechanism



## Catalog Service examples



## Objective

- **Understand the Query Process**
  - Query Request and Response messages
  - Query for Collections (Discovery) vs. Query for Granules (Inventory)
- **Working with Results**
  - Results for Collections and Granules
  - Working with large result sets
    - Iterating through few results at a time
    - Retrieving partial metadata
- **Other Catalog Service transaction**
  - How to store and execute saved queries
  - Save and present saved result sets
- **More Queries - Understanding AQL**
  - Search for each type of search criteria
  - Search using Multiple search criteria



## Setup for all the Examples

- **Log in as the user DemoUser#**
- **Run all the examples in this session while logged in**



## Example #1

- **Constructing a Query**



## Query Structure

- **Query consists of two parts**
  - XML wrapper that contains a place holder for the query and the result presentation specification and
  - IIMSAQL Query



## Result Presentation

- **The result presentation specification structure**
  - ResultType: HITS
  - Other presentation that we can look at later
- **File: 01\_ResultPresentation.xml**



## IIMSAQL Query

- **The IIMSAQL Query structure**
  - Searching for collections or granules
    - for value: collections
  - Search within which providers
    - dataCenterID: all
  - Empty where clause to list all the collections in ECHO
    - Please DO NOT use an empty where clause for granules, will attempt to return all the ECHO data
- **File: 02\_IIMSAQLQuery**
- **Validate the query in XML Spy**



## Putting it Together

- The validated IIMSAQL Query is inserted between the `<![CDATA[+++++ Insert the IIMSAQL Query here +++++]]>` in the Query message
- **File: 03\_Query.xml**



## Example #2

- **Query using different ResultTypes**
  - Experiment with ResultType RESULT\_SET\_ID, HITS and RESULTS
  - Search using Collection Query from previous example



## Query Request with RESULT\_SET\_ID

- **File: 01\_Query.xml**
  - Has ResultType as RESULT\_SET\_ID
  - Execute Query and examine the Response
    - A ResultSet is created for this query within ECHO
    - The ResultSetID for this result set is returned
    - Consists of ResultSetID only
    - Need to save this ResultSetID to retrieve results using the Present transaction



## Query Request with HITS

- **File: 01\_Query.xml**
  - Execute the same Query but this time change ResultType to HITS
  - Examine the Response
    - Consists of ResultSetID and the number of Hits
    - Probably the most useful type
    - Need to save this ResultSetID to retrieve results using the Present transaction



## Query Request with RESULTS

- **File: 01\_Query.xml**
  - Again execute the same Query changing ResultType to RESULTS
  - Examine the Response
    - Consists of ResultSetID, the number of Hits and the first few results an XML embedded as payload
    - Takes longer to return, because the Results take time to be convert to XML format
    - Need to save this ResultSetID to retrieve further results using the Present transaction



## Summary

- Executing a Query always returns a ResultSetID
- Using ResultType RESULT\_SET\_ID will not return anything else
- ResultType of HITS in addition also returns the total hits
- ResultType of RESULTS in addition returns some results
  - The number of content of the results can be modified by other options that we will look at in examples for working with large result sets



## Example #3

- Query to discover collections



## Query

- **File: 01\_IIMSACLQuery.xml**
- **IIMSACL Query**
  - for value: collections
  - dataCenterID: LPDAAC-ECS
  - Search for collections that have the 'Solar\_Azimuth\_Angle' PSA
- **Validate 01\_IIMSACLQuery.xml**
- **File: 02\_QueryRequest**
  - ResultType: RESULTS
- **Embed 01\_IIMSACLQuery.xml into 02\_QueryRequest within the CDATA section**
- **Execute Query**



## Examining Response Message

- Examine the Results only in brief, later examples will view them in more detail
- ResultSetID: RGuest6701570051035757321599 (format)
- payload
  - Presents the results that were returned
  - States what type of metadata is returned
    - CollectionMetaData



## Example #4

- Query for Granules



## Query

- File: 01\_IIMSACLQuery.xml
- IIMSACL Query
  - for value: granules
  - dataCenterID: LPDAAC-ECS
  - Search for granules that have the Solar\_Azimuth\_Angle PSA <= 33 degrees within one the Collections returned in the previous example
- Validate 01\_IIMSACLQuery.xml
- File: 02\_QueryRequest
  - ResultType: RESULTS
- Embed 01\_IIMSACLQuery.xml into 02\_QueryRequest within the CDATA section
- Execute Query



## Examining Response Message

- Examine the Results only in brief, later examples will view them in more detail
- ResultSetID: RGuest6701570051035757321599 (format)
- payload
  - Presents the results that were returned
  - States what type of metadata you are returning
    - GranuleMetaData



## Summary

- Query can be used for a Discovery of Collections or to search for Granules within those Collections



## Example #5

- Examine results for Collections



## Collection Results

- **File: 01\_QueryCollections.xml**
  - Already Includes the IIMSACL
- **IIMSACL Query**
  - for value: collections
  - dataCenterID: collections from NSIDC-ECS and ORNL
- **Result presentation**
  - ResultType: RESULTS



## Examining Response Message

- **Execute File: 01\_QueryCollections.xml**
- **Examine results in the payload**
- **Results are an embedded XML**
  - Have their own DTD
- **Extract results (cut and paste them into XML Spy)**
- **Validate Results against its own DTD**
  - Validate using XML Spy
- **Examine structure of the results**
- **Results are divided per provider**
- **Only first 10 results are returned**
- **All the metadata about each result is returned**
- **Save the ResultSetID in a separate file (for next few examples)**



## Example #6

- **Examine results for Granules**



## Granule Results

- **File: 01\_QueryGranules.xml**
  - Already Includes the IIMSACL
- **IIMSACL Query**
  - for value: granules
  - dataCenterID: granules from ORNL
- **Result presentation**
  - ResultType: RESULTS



## Examining Response Message

- **Execute File: 01\_QueryGranules.xml**
- **Examine results in the payload**
- **Results are an embedded XML**
  - Have their own DTD (Different from that for Collections)
- **Extract results (cut and paste them into XML Spy)**
- **Validate Results against its own DTD**
  - Validate using XML Spy
- **Examine structure of the results**
- **Results are divided per provider**
- **Only first 10 results are returned**
- **All the metadata about each result is returned**



## Summary

- **Results for Collections and Granules are similar**
- **Payload will have different XML**
  - Have different DTDs
- **Main structure is the same for granules and collections but the metadata for each result is different**
- **First 10 results are returned**
- **All the metadata for the Collection or Granule is returned**



## Example #7

- **Iterating through Large Result Sets**
  - We will use the Result Set created from Example #5



## Iterating Results

- **Results from the Collection Query had lots of hits**
- **Can only retrieve at most 2000 results at a time**
  - To reduce load on system and improve response times sometimes retrieving fewer results at a time maybe desirable



## Iterating Results Cont'd

- **Retrieve all metadata with default values for Presentation specification**
  - File: 01\_PresentDefault.xml
  - Execute the message after modifying it to use your ResultSetID from Example #5
    - ResultSetID: Result\_Set\_ID you saved
  - The response will have all the metadata for the first 10 collections and the Cursor will be set to 11



## Iterating Results Cont'd

- **If instead of the first 10 (default) we wish to retrieve first 11 results**
  - File: 02\_PresentFirst11.xml
  - Execute the message after modifying it to use your ResultSetID from Example #5
    - ResultSetID: Result\_Set\_ID you saved
    - IteratorSize: 11
    - Cursor: 1
  - The Response message will contain 11 collections and the cursor will be set to 12



## Iterating Results Cont'd

- **Retrieve next 11 results**
  - After retrieving the first 11, we will want to retrieve the next 11 results
  - File: 03\_PresentNext11.xml
  - Execute the message after modifying it to use your ResultSetID from Example #5
  - ResultSetID: Result\_Set\_ID you saved
    - IteratorSize: 11
    - Cursor: 12
  - The Response message will contain 11 collections and the cursor will be set to 23



## Iterating Results Cont'd

- **Keep modifying the Cursor and Iterator till you reach the last few results**
  - Leave this as an exercise



## Summary

- **Large result sets cannot be retrieved in one step. Need to iterate through results**
  - maximum 2000 at a time

## Example #8

- **Retrieve partial metadata**
  - Another way to reduce response time is to only retrieve the metadata that is of interest
  - Use TupleTypes to control the metadata that will be returned
  - Again we will use the ResultSetID from Example #5

## Retrieve Partial Metadata

- **Retrieve only the DataSetId for first 100 results**
  - File: 01\_PresentFirst100DataSetId.xml
  - Execute Present message modifying the file to use the Result\_Set\_ID you saved from Example 5
  - TupleType
    - attributeName: DataSetId
  - IteratorSize: 100
  - Examine the results
    - Only DataSetId for each Collection will be returned

## Retrieve Partial Metadata

- **File: 01\_PresentFirst100DataSetId.xml**
- **Execute Present message modifying the file to use the Result\_Set\_ID you saved from Example 5**
- **Add more TupleTypes to test other attributes that you may want**
  - Samples can be found by looking at the Results DTD for Collections
    - The TupleTypes are Case Sensitive, so make sure to type them exactly as they appear in the DTD
  - Few examples for Collections
    - ECHOInsertDate, Price, CatalogItemid, Temporal, Spatial, DateType
- **Validate the message in XML Spy**
- **Make sure the TupleTypes are valid for Collections**
- **Execute the modified message and retrieve results with more metadata**

## Summary

- **Use TupleTypes to control the metadata retrieved for each result**
- **Response time will be affected by the amount of metadata to be returned**

## Example #9

- **Managing Queries and Results**
- **Make sure you are logged in as Demo#**

## SaveQuery

- QueryName: CSDemoSavedQuery
- Execute File: 01\_SaveQuery.xml
- Saved Query is unique per user.



## Execute Saved Query

- File: 02\_ExecuteSavedQuery.xml
- Instead of specifying the entire query, only need to specify the name of the saved Query
- QueryName: CSDemoSavedQuery
- ResultType: HITS
- Execute File: 02\_ExecuteSavedQuery.xml
- Save the ResultSetID generated



## ListSavedQueries

- File: 03\_ListSavedQuery.xml
  - Lists the one query that has been saved for Demo#



## PresentSavedQuery

- QueryName: CSDemoSavedQuery
- File: 04\_PresentSavedQuery.xml
  - Presents the Query that has been saved



## RemoveSavedQuery

- QueryName: CSDemo
- File: 05\_RemoveSavedQuery.xml
  - Remove Saved queries that are no longer needed



## SaveResultSet

- SaveResultSet
  - ResultSetID: ResultSetId returned from 02\_ExecuteSavedQuery.xml
  - newResultSetID: CSDemoResultSetID
- File: 06\_SaveResultSet.xml



## ListSavedResults

- Execute File: 07\_ListSavedResultSets.xml



## Present a Saved Result Set

- **ResultSetID:** CSDemoResultSetID
- **File:** 08\_PresentSavedResultSet.xml
- Same as the request in previous example, only the **ResultSetId** is changed to the saved name



## RemoveSavedResultSet

- **ResultSetID:** CSDemoResultSetID
- **File:** 09\_RemoveSavedResultSet.xml



## Summary

- **The transactions to manage queries available in the CatalogService are:**
  - SaveQuery
  - Execute a Saved Query
  - ListSavedQueries
  - PresentSavedQueries
  - RemoveSavedQuery
- **The transactions to manage result sets available in the CatalogService are:**
  - SaveResultSet
  - ListSavedResultSets
  - RemoveSavedResultSets
  - Present results (saved and not saved)



## More Queries - IIMSAQL

- Search for each type of search criteria
- **Spatial Search**
- **Temporal Search**
- **Date Search**
- **Number Search**
- **Keyword Search**
- **Function search**



## Example # 10

- **Spatial Search**
  - Spatial search on granules
  - Search using IIMSPolygon



## Spatial - IIMSPolygon

- **Prepare IIMSAQL Query to search for granules in a portion of north east**
  - File: 01\_IIMSPolygonQuery.xml
  - Searching ORNL and LPDAAC-ECS providers for granules in a small spatial window on the east coast
  - ORNL uses Cartesian coordinate system and LPDAAC-ECS uses Geodetic, but this is transparent to the user
- **Validate query using XML Spy**
- **Insert IIMSAQL query into Query message**
  - File: 02\_Query.xml
  - Returns Spatial Information
- **Execute Query**
- **Note the number of Hits**



## Example # 11

- **Spatial Search**
  - Spatial search on granules
  - Search using Polygon (OGC Representation)



## Spatial - Polygon

- **Prepare IIMSAQL Query to search for granules in a portion of north east**
  - File: 01\_PolygonQuery.xml
  - Searching GSFC-TEST and ORNL-TEST providers for granules in a small spatial window on the east coast (Same spatial area as Example 10)
- **Validate query using XML Spy**
- **Insert IIMSAQL query into Query message**
  - File: 02\_Query.xml
  - Returns Spatial Information
- **Execute Query**
- **Note the number of Hits**
  - Should be same as Example 10



## Summary for Spatial Examples

- **Spatial Query can be issued using IIMS or OGC elements.**



## Example # 12

- **Temporal Search**
  - Range Search for granules



## Temporal Range Search

- **Prepare IIMSAQL Query to search for granules in**
  - File: 01\_TemporalRange.xml
  - Search NSIDC-ECS data for granules acquired on May 7, 2001
- **Validate query using XML Spy**
- **Insert IIMSAQL query into Query message**
  - File: 02\_Query.xml
  - Returns temporal information
- **Execute Query**



### Example #13

- **Date Search**
  - Search for collections inserted into ECHO in the last 10 days

### Date Search

- **Prepare IIMSAQL Query to search for granules in**
  - File: 01\_DateSearch.xml
  - Search for collections inserted into ECHO in the last 10 days
- **Validate query using XML Spy**
- **Insert IIMSAQL query into Query message**
  - File: 02\_Query.xml
  - Returns ECHOInsertDate
- **Execute Query**

### Example #14

- **Number Search**
  - Search for granules with cloud cover < 5%

### Number Search

- **Prepare IIMSAQL Query to search for granules in**
  - File: 01\_CampaignShortName.xml
  - Search ORNL-TEST provider for granules with CampaignShortName = 'OTTER'
- **Validate query using XML Spy**
- **Insert IIMSAQL query into Query message**
  - File: 02\_Query.xml
  - Returns all Campaigns
- **Execute Query**

### Example #15

- **Keyword Search**
  - Search for collections with parameter that begins with word "FROST"

### Keyword Search

- **Prepare IIMSAQL Query to search for granules in**
  - File: 01\_Parameter.xml
  - Search for collections with parameter that begins with word "FROST"
  - Search on the 4th level element in the DisciplineKeyword stack
- **Validate query using XML Spy**
- **Insert IIMSAQL query into Query message**
  - File: 02\_Query.xml
  - Returns VariableKeyword
- **Execute Query**

## Example #16

- **Function Search**
  - Search for only those granules that have day flag
  - If all granules of a provider have browse, this would result in too many hits.
  - So add additional search criteria. That's an example of multiple search criteria which will be the typical scenario
  - Search for only granules in this small spatial window on the east coast and that have day flag

## Function Search

- **Prepare IIMSAQL Query to search for granules in**
  - File: 01\_DayNightFlag.xml
  - Search for only granules in this small spatial window on the east coast and that have browse available
- **Validate query using XML Spy**
- **Insert IIMSAQL query into Query message**
  - File: 02\_Query.xml
  - Returns DayNightFlag and spatial information
- **Execute Query**

## Catalog Service Wrap-Up

- **Validate Validate Validate!**
  - Validate query in IIMSAQL before inserting it into the QueryRequest message
- **Use ECHO Resources judiciously**
- **All the Best!**

## Provider Profile Service



A mechanism for clients to discover what providers are participating in ECHO, and what capabilities they offer.



## Provider Profile Service Motivation

- **Clients need to know what Providers are participating in the system**
- **In case of problems with an order, Clients need to be able to provide contact information associated with the Provider from whom data was ordered**
  - Each order consists of some number of Provider Orders, one for each Provider from whom data was ordered
  - If the state of the Provider Order is such that the Client's user needs to contact the Provider, then the Provider Profile Service can be used to find out who to contact
- **In the future, Clients will be able to find out what transactions are supported by the Provider**
  - Quote, Submit, Cancel

## Example #1

- **Goal: To find out what providers are participating, and present the contact information for one.**
- **Step 1: Submit 01\_listAllProviders.xml**
- **Examine response to see what Providers have been configured into the Training System**

## Step 2 - Get the contact information

- Submit 02\_presentProviderProfile.xml
- Examine result for contact information

## Order Demonstration



## Example 1: The Order State

- Demonstrate the normal case to create an order, set options for order items, set user information, validate the order, and submit the order.
- Let user see how the order state changes in the process.

## Present a Catalog Item

- Present an catalog item you are going to order
  - File: 01\_PresentCatalogItem.xml

## Create an Order

- Create an order for the item presented in the last step.
  - File: 02\_CreateOrder.xml
- Record and save the order ID in the response, you are going to use this ID in the following steps.

## Examine the Order State

- Present the order
  - File: 03\_PresentOrder.xml
- Verify that the catalogItemID is what you ordered for and that the order state is NOT\_VALIDATED

## Update a Line Item

- Set options for the item
  - File: 04\_UpdateOrderLineItem.xml



## Re-Examine the Order State

- Present the order
  - File: 05\_PresentOrder.xml
- Verify that the options are what you set to, and that the order state remains NOT\_VALIDATED



## Update the User Information

- Set user information such as billing address, shipping address, contact address, etc. for the order
  - File: 06\_SetUserInfo.xml



## Re-Examine the Order State

- Present the order
  - File: 07\_PresentOrder.xml
- Verify that the user information is what you set to, and that the order state remains NOT\_VALIDATED



## Validate the Order

- Validate that the order has all the information included for the providers to process the order. This includes correct items, correct options for the items, complete user information
  - File: 08\_ValidateOrder.xml



## Re-Examine the Order State

- Present the order
  - File: 09\_PresentOrder.xml
- Verify that the order state has changed to VALIDATED



## Submit the Order

- **Submit the order to provider**
  - File: 10\_SubmitOrder.xml



## Re-Examine the Order State

- **Present the order**
  - File: 11\_PresentOrder.xml
- **Verify that the order state has changed to CLOSED since ECS providers do not guarantee update order status, order will be delivered upon completion**



## Example 2: Adding, Removing, and Updating Line Items

- **Demonstrate how to modify order Line items in an order using AddOrderLineItem, UpdateOrderLineItem, and DeleteOrderLineItem transactions**



## Create an Order

- **Create an order with two items**
  - File: 01\_CreateOrder.xml
- **Record and save the order ID in the response, you are going to use this ID in the following steps.**



## Add a Line Item

- **Add a third order item.**
  - File: 02\_AddOrderLineItem.xml



## Examine the Order

- **Present the order**
  - File: 03\_PresentOrder.xml
- **Verify that there are three order line items in your order**



## Update a Line Item

- Update the first order line item with different quantity
  - File: 04\_UpdateOrderLineItem.xml



## Re-Examine the Order

- Present the order
  - File: 05\_PresentOrder.xml
- Verify that the quantity ordered in the first item is the same as what you specified in the last step



## Remove a Line Item

- Delete the second order line item from the order
  - File: 06\_DeleteOrderLineItem.xml



## Re-Examine the Order

- Present the order
  - File: 07\_PresentOrder.xml
- Verify that the second order line item has been deleted.



## Example 3: Order State and How It Is Affected by Modification of an Order Line Item

- Demonstrate what happen if you modify order Line items in an order after the validation of the order



## Create an Order

- Create an order with one item
  - File: 01\_CreateOrder.xml
- Record and save the order ID in the response, you are going to use this ID in the following steps.



## Modify a Line Item

- **Set options to the order line item.**
  - File: 02\_UpdateOrderLineItem.xml



## Set the User Information

- **Set the user information for the order**
  - File: 03\_SetUserInformation.xml



## Validate the Order

- **Validate the order**
  - File: 04\_ValidateOrder.xml



## Examine the Order

- **Present the order**
  - File: 05\_PresentOrder.xml
- **Verify that the order is VALIDATED**



## Add a Line Item

- **Modify the validated order by adding a new item**
  - File: 06\_AddOrderLineItem.xml



## Re-Examine the Order

- **Present the order**
  - File: 07\_PresentOrder.xml
- **Verify that the order state changed back to NOT\_VALIDATED after addition of an item**



## Submit the Order

- **Submit the order**
  - File: 08\_SubmitOrder.xml
- **Verify that you receive an error message in the response, because you can not submit a not-validated order**



## Modify a Line Item

- **Set options for the newly added item**
  - File: 09\_UpdateOrderLineItem.xml



## Validate the Order

- **Validate the order**
  - File: 10\_ValidateOrder.xml
- **Verify that the order is in the VALIDATED state after successfully validated.**



## Modify a Line Item

- **Modify a validated order by updating the quantity of the first item (options can be included in the line item, so you don't have to use a separate step for setting options)**
  - File: 11\_UpdateOrderLineItem.xml



## Re-Examine the Order

- **Present the order**
  - File: 12\_PresentOrder.xml
- **Verify that the order state has changed back to NOT\_VALIDATED after update of an item**



## Re-Validate the Order

- **Validate the order**
  - File: 13\_ValidateOrder.xml
- **Verify that the order is in the VALIDATED state after successfully validated.**



## Remove a Line Item

- Delete the first item from the order
  - File: 14\_DeleteOrderLineItem.xml



## Re-Examine the Order

- Present the order
  - File: 15\_PresentOrder.xml
- Verify that the order state has changed back to NOT\_VALIDATED after deletion of an item



## Example 4: Managing Provider Orders

- Demonstrate how to use provider order level transactions to manage the provider orders in an order
- Help understanding of the difference between an order state and a provider order state



## Create an Order

- Create an order with two items, each from a different provider. (Options are included in the creation time)
  - File: 01\_CreateOrder.xml
- Record and save the order ID in the response, you are going to use this ID in the following steps.



## Examine the Order

- Present the order.
  - File: 02\_PresentOrder.xml
- Check the provider order states to verify that both provider orders in NOT\_VALIDATED state.
- Check the order state to the entire order in NOT\_VALIDATED state
- Record and save the provider Id in the provider order section in the response, you are going to use this id in the following steps



## Examine a Provider Order

- Present the first provider order.
  - File: 03\_PresentProviderOrder.xml
- Verify that only the provider order you specified is presented



## Set the User Information

- **Set the user information for the order**
  - File: 04\_SetUserInformation.xml



## Validate the Order

- **Validate the order**
  - File: 05\_ValidateOrder.xml



## Re-Examine the Order

- **Present the order**
  - File: 06\_PresentOrder.xml
- **Check the provider order states to verify that both provider orders in VALIDATED state.**
- **Check the order state to the entire order in VALIDATED state**



## Remove a Provider Order

- **Delete the first provider order from the order**
  - File: 07\_DeleteProviderOrder.xml



## Re-Examine the Removed Provider Order

- **Present the provider order just deleted**
  - File: 08\_PresentProviderOrder.xml
- **Verify that you receive an error message in the response, because the provider order no longer exists in the order**



## Re-Examine the Order

- **Present the order**
  - File: 09\_PresentOrder.xml
- **Verify that only one provider order left in the order**
- **Verify that the provider order state remains VALIDATED and order state remains VALIDATED**



## Add a Line Item

- Add another item belonging to a provider that is different with the one already in the order
  - File: 10\_AddOrderLineItemToOrder.xml



## Re-Examine the Order

- Present the order
  - File: 11\_PresentOrder.xml
- Verify that there are now two provider orders in the order
- Verify that the newly created provider order is in NOT\_VALIDATED state and the other provider order remains in VALIDATED state
- Verify that the entire order is in NOT\_VALIDATED state, check your user's guide for more information about how order state is determined from provider order states



## Remove a Line Item

- Delete the order item you just created
  - Note that this is the only item in this provider order
  - File: 12\_DeleteOrderLineItem.xml



## Re-Examine the Order

- Present the order
  - File: 13\_PresentOrder.xml
- Verify that there is only one provider order in the order because you deleted the only item in a provider order and hence the provider order is deleted.
- Verify that the provider order still in the order remains in VALIDATED state
- Verify that the entire order state changed to VALIDATED state since you have removed the NON\_VALIDATED one.



## Example 5: Ordering Restricted Metadata

- Demonstrate how changes in restriction on ordering a catalog item by the provider affect your order



## Create an Order

- Create an order based on catalog item G974622-GSFC-TEST that belongs to provider GSFC-TEST
  - File: 01\_CreateOrder.xml
- Record and save the order ID in the response, you are going to use this ID in the following steps.



## Update a Line Item

- Update the quantity for item G974622-GSFC-TEST
  - File: 02\_UpdateOrderLineItem.xml
- Verify that the update is allowed



## Watch Me - Do NOT Do This

- I will now switch users and restrict the granule
- You watch as I:
  - Log In as the Admin User(User101) and setProviderContext
  - Create a condition and a restriction
  - File: 03\_CreateCondition.xml
  - File: 04\_CreateRestriction.xml



## Update the Line Item

- Update the quantity of an order line item
- Verify that you received an error message that the item is no longer orderable because of the new restriction you just set in the last step
- File: 05\_UpdateOrderLineItem.xml



## Watch Me - Do NOT Do This!

- I will now remove the restriction on the granule
- Watch as I:
  - Log in as the Admin User(User101) and setProviderContext
- Delete the restriction and condition on ordering item G974622-GSFC-TEST
  - File: 06\_DeleteRestriction.xml
  - File: 07\_DeleteCondition.xml



## Update the Line Item

- Update the quantity of the item G974622-GSFC-TEST
- Verify that the update is allowed because the restriction has been deleted
- File: 08\_UpdateOrderLineItem.xml



## Example 6: Managing Multiple Orders

- Demonstrate handling of multiple orders



## Create a New User and Login

- Create a user OrderUser# that can access transaction ListUnsubmittedOrderSummary used in the following steps
  - File: 01\_CreateUser.xml
- Log in the user just created
  - User name: OrderUser#
  - Password: Password101



## Create an Order

- Create the first order
  - File: 02\_CreateOrder.xml
- Record and save the order ID #1 in the response, you are going to use this ID in the following steps.



## Create Another Order

- Create the second order
  - File: 03\_CreateOrder.xml
- Record and save the order ID #2 in the response, you are going to use this ID in the following steps.



## One More Order . . .

- Create the third order
  - File: 04\_CreateOrder.xml
- Record and save the order ID #3 in the response, you are going to use this ID in the following steps.



## Present the Orders

- Present multiple orders
  - File: 05\_PresentOrder.xml



## List UnSubmitted Orders

- List Unsubmitted orders
  - File: 06\_ListUnsubmittedOrderSummary.xml
- Verify that all three orders are listed



## Delete an Order

- Delete the first and the second orders
  - File: 07\_DeleteOrder.xml



## Re-List UnSubmitted Orders

- List Unsubmitted orders
  - File: 08\_ListUnsubmittedOrderSummary.xml
- Verify that only one order is listed after you deleted two orders
- Log out from OrderUser#



## Subscription Service examples



## Objective

- Issue a Query to determine the collection metadata and the granule metadata information required for a subscription
- Create subscriptions
  - Subscriptions will be created to a collection in ECHO
  - These Subscriptions will subscribe to the collection metadata updates and granule metadata updates
- One person will emulate a Provider sending new data to the system with a few granules in the relevant dataset as well as an update to the collection metadata
- Verify subscription update email was received
- Clean-Up



## Example #1

- Subscribing to collection metadata



## Issue a Query

- Look specifically for the 'Equally distributed insert time V001' collection
- You should receive the collection metadata
- File: 01\_Query.xml



## Create Subscription

- **SubscriptionName:** Temporal Collection Sub for Demo0
  - **providerName:** DEMO\_PROVIDER#
  - **datasetName:** Equally distributed insert time V001
  - **MetadataFilterInfo**
    - spatialCondition: GLOBAL
  - **MetadataActionFilter**
    - DTDType: ECHO
    - CompressionType: UNCOMPRESSED
    - SubscriptionUpdateType: COLLECTIONS\_ONLY
  - **DeliveryInfo**
    - DeliveryType: EMAIL
    - DeliveryAddress: demo\_provider#@yahoo.com
    - limitSize: 3.0
  - **StopDate:** Dec. 30, 2003
- **File:** 02\_CreateSubscription.xml (make sure change email address correspondingly).



## Ingest Metadata

- An ingest will now be performed for a provider sending a collection metadata update into ECHO
  - **Take 03\_Ingest.xml** and FTP it to [orval.gsfc.nasa.gov](http://orval.gsfc.nasa.gov)
    - Open a dos window
    - Type this into the URL:
      - Cd c:\the file path
      - ftp orval.gsfc.nasa.gov
      - Type demo\_provider# as User, anything as password
    - Navigate to this directory:
      - Cd data/collection
      - Put 03\_Ingest.xml
    - The Ingest script will automatically pick it up within 5 minutes
- **File:** 03\_Ingest.xml



## Verify Subscription

- **Login to your yahoo email account**
- **An e-mail message will be delivered shortly**
  - Subject Line: Subscription Data Delivery for Temporal Collection Sub for Demo0
  - Content
    - Dear User,
    - Attached please find the Collection metadata update on Equally distributed Insert time V001 received from DEMO\_PROVIDER# provider to which you subscribed.
    - ECHO Subscription Service
    - Email: [echosubscription@gsst.com](mailto:echosubscription@gsst.com)
    - Phone: 301 474 9696
- **Open the attached file and view the CollectionDescription field**
  - It should be different than the CollectionDescription from the Query
  - You can verify it against your Worksheet



## Clean-Up

- **Delete your subscription**
    - SubscriptionName: Temporal Collection Sub for Demo0
- **File:** 03\_DeleteSubscription



## Summary

- We located a collection to which we would like to subscribe
- We created a subscription to that collection to notify us whenever a collection metadata update occurs for that collection
- We examined the email update that is generated when a metadata update occurs
- We deleted the subscription



## Example #2

- Subscribing to granule metadata



## Issue a Query

- Look specifically for the granules belonging to the 'Equally distributed insert time V001' collection
- You should receive the granule Equally distributed insert time: granule #0: 2000-01-01 00:00:00.000 metadata
- File: 01\_Query.xml



## Create Subscription

- MetadataActionFilter
  - SubscriptionUpdateType - GRANULES\_ONLY
- File: 02\_CreateSubscription



## Ingest Metadata

- An ingest will now be performed for a provider sending a granule metadata update into ECHO
- Take 03\_Ingest.xml and FTP it to orval.gsfc.nasa.gov
  - Open a dos window
  - Type this into the URL:
    - Cd c:\the file path
    - ftp orval.gsfc.nasa.gov
    - Type demo\_provider# as User , anything as password
  - Navigate to this directory:
    - Cd data/collection
    - Put 03\_Ingest.xml
  - The Ingest script will automatically pick it up within 5 minutes
- File: 03\_Ingest.xml



## Verify Subscription

- Stay logged into your yahoo email account
- Another email is on the way
- Subject Line
  - Subscription Data Delivery for Temporal Collection Sub for Demo0
- Email Content
  - Dear User,  
Attached please find Metadata update of 1 granules on Equally distributed insert time V001 received from DEMO\_PROVIDER# provider to which you subscribed.  
ECHO Subscription Service  
Email: [echosubscription@gst.com](mailto:echosubscription@gst.com)
  - Phone: 301 474 9696
- Open the attached file
- Search for the Granule 'Equally distributed insert time: granule #0: 2000-01-01 00:00:00.000'



## Clean-Up

- Delete your subscription
  - SubscriptionName: Temporal Collection Sub for Demo0
- File: 04\_DeleteSubscription



## Summary

- We located a granule to which we would like to subscribe
- We created a subscription to that granule to notify us whenever a granule metadata update occurs
- We examined the email update that is generated when a metadata update occurs
- We deleted the subscription



### Example #3

- **Subscribing to both collection metadata and granule metadata**



### Create Subscription

- **MetadataActionFilter**
  - SubscriptionUpdateType - BOTH
- **File: 01\_CreateSubscription**



### Ingest Metadata

- An ingest will now be performed for a provider sending a collection metadata update and granule metadata update of the 'Equally distributed insert time V001' collection into ECHO
- **FTP 02\_Ingest.xml, 03\_Ingest.xml to orval.gsfc.nasa.gov**
  - Open a dos window
  - Type this into the URL:
    - Cd c:\the file path
    - ftp orval.gsfc.nasa.gov
    - Type demo\_provider# as User , anything as password
  - Navigate to this directory:
    - Cd data/collection
    - Put 02\_Ingest.xml
    - Cd ../granule
    - Put 03\_Ingest.xml
  - The Ingest script will automatically pick it up within 5 minutes
- Collection File: 02\_Ingest.xml
- Granule File: 03\_Ingest.xml



### Verify Subscription - Collection

- **Stay logged into your yahoo email account**
- **Another email is on the way**
- **Subject Line**
  - Subscription Data Delivery for Temporal Collection Sub for Demo0
- **Email Content**
  - Dear User,  
Attached please find the Collection metadata update on **Equally distributed insert time V001** received from DEMO\_PROVIDER# provider to which you subscribed.  
ECHO Subscription Service  
Email: [echosubscription@gst.com](mailto:echosubscription@gst.com)  
- Phone: 301 474 9696
- **Open the attached file and view the CollectionDescription field**
  - It should have reverted to the original value
    - Check your Worksheet



### Verify Subscription - Granule

- **Another email should be in your inbox**
- **Subject Line**
  - Subscription Data Delivery for Temporal Collection Sub for Demo0
- **Email Content**
  - Dear User,  
Attached please find Metadata update of 1 granules on **Equally distributed insert time V001** received from DEMO\_PROVIDER1 provider to which you subscribed.  
ECHO Subscription Service  
Email: [echosubscription@gst.com](mailto:echosubscription@gst.com)  
- Phone: 301 474 9696
- **Search for the Granule** 'Equally distributed insert time: granule #0: 2000-01-01 00:00:00.000'



### Clean-Up

- **Delete your subscription**
  - SubscriptionName: Temporal Collection Sub for Demo0
- **File: 04\_DeleteSubscription**



## Summary

- We located a collection and a granule to which we would like to subscribe
- We created a subscription to that collection and granule to notify us whenever a collection or granule metadata update occurs
- We examined the email update that is generated when a metadata update occurs
- We deleted the subscription



## Example #4

- Subscribing to all collection metadata for all ECHO providers



## Create Subscription

- Here we will use wildcards (\*) to create a subscription that will notify the user whenever a metadata update ingest has been performed on any collection belonging to any provider
- **SubscriptionName:** All Collections for Demo0
- **providerName:** \*
- **datasetName:** \*
- **MetadataActionFilter**
  - SubscriptionUpdateType: ALL\_COLLECTIONS
- **File:** 01\_CreateSubscription.xml



## Ingest Metadata

- An ingest will now be performed for several providers sending collection metadata update into ECHO
- **FTP 02\_Ingest.xml, 03\_Ingest.xml** to `orval.gsfc.nasa.gov`
  - Open a dos window
  - Type this into the URL:
    - Cd c:\the file path
    - ftp orval.gsfc.nasa.gov
    - Type demo\_provider# as User , anything as password
  - Navigate to this directory:
    - Cd data/collection
    - Put 02\_Ingest.xml
    - Cd ../granule
    - Put 03\_Ingest.xml
  - The Ingest script will automatically pick it up within 5 minutes



## Verify Subscription

- Two Emails will be delivered to your yahoo email account
- Both Emails will be alike, except for the attachments
- **Subject Line**
  - Subscription Data Delivery for All Collections for Demo#
- **Email Content**
  - Dear User,
  - Attached please find the Collection metadata update on \* received from \* provider to which you subscribed.
  - ECHO Subscription Service
  - Email: [echosubscription@gst.com](mailto:echosubscription@gst.com)
  - Phone: 301 474 9696
- **Note:** The Provider's metadata update did not actually update the collection. This was just a demonstration to show how you can receive any collection that has been updated by any Provider of ECHO



## Clean-Up

- **Delete your subscription**
  - SubscriptionName: All Collections for Demo0
- **File:** 04\_DeleteSubscription



## Summary

- We created a subscription using wildcards to notify us whenever a collection is updated by any provider of ECHO
- We deleted the subscription



## Example #5

- Subscribing to metadata that is restricted by a provider



## The Plan

- In the DataManagement Service hands-on, you learned how to restrict access to a collection of a Provider
- These restrictions apply to subscriptions as well
- In this exercise, we will create a condition and a restriction on a piece of metadata
- We will then create a subscription to that metadata to notify us of an update
- We will examine what a subscription email looks like when a restriction has been placed on that metadata



## Create Condition

- SetProviderContext as DEMO\_PROVIDER#
- Create a Boolean 'True' Condition
- Name: true condition
- BooleanFlag: True
- File: 01\_CreateBooleanCondition.xml



## Create Restriction

- ActionType: View
- ConditionType: Boolean
- ConditionName: true condition
- Comparator: EQUALS
- DataType: COLLECTION
- DataValue: Equally distributed insert time V001
- File: 02\_CreateRestriction.xml



## Create Subscription

- Logout from DEMO\_PROVIDER#
- Login to Demo#
- Create a Subscription
  - SubscriptionName: RESTRICT\_TEMPORAL Sub for Demo0
  - providerName: DEMO\_PROVIDER#
  - DatasetName: Equally distributed insert time V001
  - MetadataActionFilter
    - SubscriptionUpdateType: COLLECTIONS\_ONLY
- File: 03\_CreateSubscription.xml



## Ingest Metadata

- An ingest will now be performed by a provider sending collection and granule metadata update into ECHO for ingestion
- Take `04_Ingest.xml` and FTP it to `orval.gsfc.nasa.gov`
  - Open a dos window
  - Type this into the URL:
    - `Cd c:\the file path`
    - `ftp orval.gsfc.nasa.gov`
    - Type `demo_provider#` as User , anything as password
  - Navigate to this directory:
    - `Cd data/collection`
    - Put `04_Ingest.xml`
  - The Ingest script will automatically pick it up within 5 minutes
- File: `04_Ingest.xml`



## Verify Subscription

- Stay logged into your yahoo email account
- Another email is on the way
- Subject Line
  - Subscription Data Delivery for RESTRICT\_TEMPORAL Sub for Demo0
  - Email Content
- Dear user,  
Attached please find Collection metadata update on Equally distributed insert time V001 received from DEMO\_PROVIDER# provider to which you subscribed.  
ECHO Subscription Service  
Email: [echosubscription@gst.com](mailto:echosubscription@gst.com)  
- Phone: 301 474 9696
- You will find a message indicating that the collection you subscribed to has been restricted



## Clean-Up

- Delete your subscription
  - SubscriptionName: RESTRICT\_TEMPORAL Sub for Demo0
- SetProviderContext as DEMO\_PROVIDER#
- Delete your Restriction
- Delete your Condition
- Logout from DEMO\_PROVIDER#
- File: `05_DeleteSubscription.xml`
- File: `06_DeleteRestriction.xml`
- File: `07_DeleteCondition.xml`



## Example #6

- Managing a Subscription



## Create Subscription

- SubscriptionName: RESTRICT\_TEMPORAL Sub for Demo0
- providerName: DEMO\_PROVIDER#
- datasetName: Equally distributed insert time V001
- MetadataFilterInfo
  - spatialCondition: GLOBAL
- MetadataActionFilter
  - DTDTType: ECHO
  - CompressionType: UNCOMPRESSED
  - SubscriptionUpdateType: COLLECTIONS\_ONLY
- DeliveryInfo
  - DeliveryType: EMAIL
  - DeliveryAddress: `demo_provider#@yahoo.com`
  - limitSize: 3.0
- StopDate: Dec. 30. 2003
- File: `01_CreateSubscription.xml`



## ListSubscriptionNames

- Lists the names of all existing metadata subscriptions subscribed by the user
- File: `02_ListSubscriptionNames.xml`



## PresentSubscription

- Presents info on an existing metadata subscription
- **SubscriptionName:** RESTRICT\_TEMPORAL Sub for Demo0
- **File:** 03\_PresentSubscription.xml



## UpdateSubscription

- Updates an existing Metadata Subscription
- **SubscriptionName:** RESTRICT\_TEMPORAL Sub for Demo0 (current name of subscription)
- **MetadataSubscription**
  - SubscriptionName: RESTRICT\_TEMPORAL Sub for Demo0
  - ProviderName: DEMO\_PROVIDER#
  - dataSetName: Equally distributed insert time V001
  - MetadataFilterInfo
    - spatialCondition: GLOBAL
  - MetadataActionInfo
    - DTDType: ECHO
    - CompressionType: UNCOMPRESSED
    - SubscriptionUpdateType: COLLECTIONS\_ONLY
  - DeliveryInfo
    - DeliveryType: EMAIL
    - DeliveryAddress: demo\_provider@yahoo.com
    - limitSize: 3.0
    - StopDate: Dec. 30, 2003
- **File:** 04\_UpdateSubscription.xml



## PauseSubscription

- Pauses an existing metadata subscription
- **SubscriptionName:** RESTRICT\_TEMPORAL Sub for Demo0
- **File:** 05\_PauseSubscription.xml



## ResumeSubscription

- Resumes a paused metadata subscription
- **SubscriptionName:** RESTRICT\_TEMPORAL Sub for Demo0
- **File:** 06\_ResumeSubscription.xml



## Clean-Up

- Delete your subscription
  - SubscriptionName: RESTRICT\_TEMPORAL Sub for Demo0
- **File:** 07\_DeleteSubscription



## Summary

- We created a subscription
- Effectively managed the subscription



## Subscription Service Wrap-Up

- Subscriptions provide a way for users of ECHO to keep informed about recent changes made to collections and/or granules of interest
- Subscriptions can be applied to just a collection or a granule, collections and their granules, or all collections for all providers of ECHO
- Datamanagement Service Rules apply to subscriptions
- Subscriptions can be managed effectively using numerous transactions located in the Subscription Service