Example SOAP Client

Introduction

The Fedora API-A interface is implemented as a SOAP service that consists of the following methods:

- **DescribeRepository** – gets information about the repository, including version
- **GetDatastreamDissemination** – gets the content's of the specified datastream
- **GetDissemination** – gets a dissemination result
- **GetObjectHistory** – gets the change history of an object consisting of a list of timestamps indicating when object components were created or modified
- **GetObjectProfile** – gets object profile
- **FindObjects** – gets the requested ObjectFields on all objects in the repository matching the given criteria
- **ListDatastreams** – lists the datastreams in the specified object
- **ListMethods** – lists the methods in the specified object
- **ResumeFindObjects** – gets the next list of results from a truncated findObjects response

For more information on the method definitions, refer to the API descriptions located at Fedora Repository Web Service Interfaces.

The Fedora API-A sample SOAP client is an example of a java servlet-based client that provides a front end to the Fedora Access API-A SOAP service. The sample client is designed to provide a *browser centric* view of the Fedora Access interface. Return types from the Fedora Access SOAP service are translated into a form suitable for viewing with a web browser; i.e., MIME-typed streams. Applications that can readily handle SOAP requests and responses would most likely communicate directly with the Fedora Access SOAP service rather than use a java servlet as an intermediary. *This servlet is provided as an example of how to construct a client that uses the Fedora Access API via SOAP and currently does not implement all methods found in API-A. As noted below, the sample Soap Client does not provide implementations for the methods FindObjects and ResumeFindObjects.*

Client Syntax

Input parameters for the servlet include:

- **action_** – name of Fedora service which must be one of the following:
  - *DescribeRepository* – gets description of the repository.
    - no parameters required
  - *GetDatastreamDissemination* – gets the contents of the specified datastream.
    - requires PID_
    - requires dsID_
  - *GetDissemination* – gets a dissemination result.
    - requires PID_
    - requires bDefPID_
    - requires methodName_
  - *GetObjectHistory* – gets object change history.
    - requires PID_
  - *GetObjectProfile* – gets object profile.
    - requires PID_
  - *FindObjects* – gets the requested ObjectFields on all objects in the repository matching the given criteria. ***(NOT CURRENTLY IMPLEMENTED)***
  - ListDatastreams – lists the datastreams in the specified object.
    - requires PID_
  - *ListMethods* – lists the methods for the specified object.
    - requires PID_
  - *ResumeFindObjects* – gets the next list of results from a truncated findObjects response. ***(NOT CURRENTLY IMPLEMENTED)***
- **PID_** – persistent identifier of the digital object
- **bDefPID_** – persistent identifier of the Service Definition object
- **methodName_** – name of the method
- **dsID_** – datastream ID
- **asOfDateTime_** – versioning datetime stamp (optional). Proper syntax is YYYY-MM-DDTHH:MM:SS where HH is 24-hour clock.
- **xml_** – boolean switch used in conjunction with GetBehaviorDefinitions, GetBehaviorMethods, GetObjectMethods, GetObjectProfile and DescribeRepository methods that determines whether output is formatted as XML or as HTML; value of "true" indicates that the results are to be returned as XML; value of false or omission indicates display in HTML table format. (optional)
- **userParms** – service methods may require or provide optional parameters that may be input as arguments to a service method; these parameters are entered as name/value pairs like the other serlvet parameters (optional)

Note that all servlet parameter names that are implementation specific end with the underscore character ("_"). This is done to avoid possible name clashes with user-supplied method parameter names. As a general rule, user-supplied parameters should never contain names that end with the underscore character to prevent possible name conflicts.

Configuration

The sample client can be run as an integral part of the Fedora server or as a standalone client on a different machine or server. If running outside the Fedora server, its only requirements are a servlet engine using Java Servlet API 2.4 or higher, the endpoint of a functional Fedora Server, and the servlet path mapping used to access the soap client. The endpoint and servlet path info are configured in the `soapclient.properties` file that is located in the WEB-INF directory of the soapclient webapp module in the Fedora distribution directory tree. The properties file specifies three variables:

- **fedoraEndpoint** – the URL of the Fedora API-A SOAP service; default is "http://localhost:8080/fedora/services/access"

- **soapClientServletPath** – the servlet path used to access the soapclient; default is "/soapclient/apia"
- **soapClientMethodParmResolverServletPath** – the servlet mapping used to access the MethodParmResolverServlet which is a utility servlet of the soap client; default is "/soapclient/getAccessParmResolver"

The web.xml file for the soap client webapp uses these default settings. If the webapp is ported to another servlet engine and any of the serlvet mappings change, the web.xml and the soapclient.properties file must both be updated accordingly. The servlets will dynamically get the values in the soapclient. properties file so no changes should be needed in the source code.

When the Fedora server is built, it will automatically create the soapclient webapp module and place it in the Fedora distribution directory. The generated SOAP type libraries that are bundled with the client are tied to the WSDL type definitions defined for Fedora API-A. Any changes made to the WSDL for API-A will require rebuilding of the Fedora server and the soap client webapp module.

Note that if you change the host name or port number on which the Fedora server is running, you must also update the value for the fedoraEndpoint parameter in the soapclient.properties file to reflect the new values. Otherwise, the soap client will be unable to connect to the Fedora server running under the new host name or port number. If you are running the soap client bundled together with the Fedora server, you need to stop the Fedora server, update the soapclient.properties file, and then restart the Fedora server so the new property values will be recognized by the soapclient webapp.

Examples

Get information about the repository:

http://localhost:8080/soapclient/apia?action_=DescribeRepository

Get the thumbnail datastream (with DSID of DS1) in data object with a PID of demo:5

http://localhost:8080/soapclient/apia?action_=GetDatastreamDissemination&PID_=demo:5&dsID_=DS1

Get the Dissemination for a data object with a PID of demo:5 and associated behavior definition object with a PID of demo:1 and methodName of getThumbnail:

http://localhost:8080/soapclient/apia?action_=GetDissemination&PID_=demo:5&bDefPID_=demo:1&methodName_=getThumbnail

Get the ObjectHistory for a data object with a PID of demo:5:

http://localhost:8080/soapclient/apia?action_=GetObjectHistory&PID_=demo:5

Get the ObjectProfile for a data object with a PID of demo:5:

http://localhost:8080/soapclient/apia?action_=GetObjectProfile&PID_=demo:5

List the datastreams for data object with PID of demo:5

http://localhost:8080/soapclient/apia?action_=ListDatastreams&PID_=demo:5

List the methods for data object with PID of demo:5

http://localhost:8080/soapclient/apia?action_=ListMethods&PID_=demo:5