

1. Introduction

With the developments in the different digital repositories, evolved different set of Metadata structures to implement the digital repositories. The increase in different Metadata standards gave birth to a question, “How to connect different digital repositories that have different Metadata structures?”. ECL protocol was developed as an answer to this question. ECL allows to share Metadata between different repositories which could have same or different Metadata structures. For e.g. ECL would allow users from one digital repository that uses IMS LOM Metadata structure to search and retrieve Metadata from another repository that uses Dublin Core Metadata format.

ECL implements the IMS Digital Repository Interoperability specifications (for search , submit, gather, subscribe and request). ECL uses messaging system to connect different digital repositories. Different ECL messages are placed in the SOAP body and passed on to the target Digital Repository over http. The ECL connected repositories when receive the request, converts the request to repository's native search query, retrieve the results and send them in ECL standard response format.

With the development of the ECL, there rise another question, “How to provide protected access to Metadata in one repository to another repository?”. If some repository wants to provide confidential Metadata, the current ECL protocol doesn't have a mechanism to provide a secure environment for the confidential Metadata. ECL Security project is developed to address this question. ECL Security is an extension to the existing ECL protocol. Using ECL Security, repositories could search/exchange confidential Metadata, and make authorization decisions for individual requests.

2. Possible Approaches

Like many possible approaches in Metadata structures for digital repository implementation, there are many possible approaches to address the security requirement in ECL. For e.g. setting up a SSL connection based on trust and then sending the Metadata, or Encrypting the SOAP message itself to send the Metadata. The basic goal during brainstorming was to support wide variety of security mechanisms, with a side requirement of wide acceptance.

Out of many possible solutions, OASIS Web Services Security drafts satisfied most of the requirements. The security drafts from OASIS for Web Service Security were widely accepted, provides support for wide variety of security mechanisms, and provides a systematic approach for the security requirement. The OASIS family of protocol specifications is still underdevelopment and not yet finished. Once completed the family of protocol would provide a complete security solution. The main purpose for OASIS Web Services Security drafts is to enable applications to construct secure SOAP message exchanges. The set of specifications is intended to provide a flexible set of mechanisms that can be used to construct a range of security protocols.

3. Design Goals

ECL Security project has the following basic designing goals :

- Provide extension to existing ECL protocol to incorporate different security mechanisms.
- Comply with the OASIS Web Services Security drafts : SOAP Security, UserName Token, SAML Profile etc.
- Simplify the integration of security.

4. Solution

WSS4J is a project under development at Apache. It provides implementation of different OASIS Web Service Security drafts. WSS4J uses Apache Axis and Apache XML-Security projects and inter-operates with JAX-RPC based server/client and .NET server/clients.

WSS4J uses Axis handlers to address the security requirements. The WSS4J handlers add/change/verify the security information in the SOAP message. WSS4J handlers, based on the configuration file on client side, adds the security information in the SOAP message, and transmits it further to Axis higher level handlers. When the client calls the service, the client side deployment file specifies WSS4J handler. The client configures the security parameter in the “Call” object; WSS4J handler retrieves those parameters and

adds the security information in the SOAP message.

At the server side, the Web Service is deployed providing WSS4J as one of the handler, which could takes few arguments. The WSS4J handler tries to look for the required arguments in the Axis “MessageContext” object first, if not, than looks into the deployment arguments. When the request arrives at the server side, Axis after basic processing, passes the SOAP message to the WSS4J handler, which verifies the security information based on the parameters (either from the “MessageContext” object or deployment parameters). If the security verification is successful it passes the SOAP message to further handlers, or to the service end point implementation.

The server-side implementation of ECL Security using WSS4J has two choices, either direct clients to use the static configuration (based on configuration files), or allow a little flexibility using the arguments passed in the “MessageContext” object. The second choice even though not completely rid of configuration file is clearly a better choice to provide security implementation in ECL security. The ECL security provides “ECL Security Handlers” on the top of WSS4J handlers, which adds the security parameters in the “MessageContext” object. The WSS4J handlers retrieves the security parameters from the “MessageContext” object and process them.

The ECL Security provides API on the client side to add the security parameters. For e.g.

```
ECLUserNameToken userToken = new ECLUserNameToken();  
userToken.setUsername(“username”);
```

The ECL client configures the SecurityToken and executes the call, as per the normal ECL client call, but with additional parameter, i.e. SecurityToken(e.g. userToken).

The ECL repository server, when receives the request, first passes the handle to “ECL Security handler”. The ECL Security handler, using a configuration file, adds the parameters in “MessageContext” object. WSS4J extracts those parameters and processes the SOAP security. Providing ECL Security server side handler with configuration file provides flexibility of changing the security mechanism implementation without redeploying the service. Also, there are further scope of implementing multiple security

mechanism for single service. The current version has a limitation of one security mechanism per service (for e.g. UserNameToken, or SAML profile etc.). The ECL services will have choice of deciding on which security mechanism to support, but can provide only one security mechanism at a time.