Digital Repository Interoperability: Design, Implementation and Deployment of the ECL Protocol and Connecting Middleware

Timmy Eap
Simon Fraser University
Surrey 2400 Central City
Surrey, BC, Canada, V3T 2W1
teap@sfu.ca

Marek Hatala
Simon Fraser University
Surrey 2400 Central City
Surrey, BC, Canada, V3T 2W1
mhatala@sfu.ca

Griff Richards
BCIT Technology Centre &
SFU Surrey
Surrey, BC, Canada, V3T 2W1
griff@sfu.ca

ABSTRACT
This paper describes the design and implementation of the eduSource Communication Layer (ECL) protocol. ECL is one outcome of a pan-Canadian project called eduSource Canada to build an open network of interoperable digital repositories. The design goal was to achieve a highly flexible, easy-to-use, and platform independent communication layer protocol that allows new and existing repositories to communicate and share resources across a network. ECL conforms to IMS Digital Repository Interoperability (DRI) specifications and supports four main functions: search/expose, submit/store, gather/expose and request/deliver. The ECL protocol builds on the latest standards and is flexible with respect to metadata schemas and repository contents. To support easy adoption of the protocol we provide middleware components for connecting existing systems. The ECL is currently used in the eduSource network, and we have begun work bridging with other interoperable initiatives such as Open Knowledge Initiative (OKI). Based on our experience, ECL is truly flexible and easy to use.

Categories and Subject Descriptors
D.2.12 [**Software engineering**] Interoperability

General Terms
Design, Standardization

Keywords
Interoperability, Protocols, Middleware

1. INTRODUCTION
Web technology in education and training has generated several centralized digital learning object repositories in the last few years. By itself, a single repository is insufficient to provide an adequate knowledge base for a learner who wants and deserves a global body of knowledge. This can only be done through the joining of all digital repositories to create a network of knowledge. To promote this sharing and reuse of learning objects projects such as SMETE (www.smete.org/, Merlot (www.merlot.org), EdNa (www.edna.edu.au), and RDN (www.rdn.ac.uk) have put up services enabling individuals and other repositories to search and retrieve metadata from their repositories. Consequently, there is an emergence of new protocols. Rather than leaving application developers dealing with these emerging protocols, eduSource Canada took a first step towards improving interoperability by developing a standard communication protocol (ECL) that is easy to integrate with existing repositories and is highly interoperable. This was not an easy task as repositories are implemented on different platforms and use different metadata schemas. An easy but inflexible solution would be to tie the ECL with web services technology and enforce IEEE LOM as a metadata standard. Instead, we designed the ECL as a neutral protocol that can accommodate different metadata schemas. Repositories can simply list their preferred schemas on the UDDI registry and continue to service only their preferred schemas. Clients can use XSLT to convert metadata to the schemas that they can work with. To help developers, we provide a set of XSLT style sheets that optimize the conversion of metadata and minimize the potential data loss.

The pilot implementation of the ECL is on Java platform and uses SOAP as a messaging protocol. Implementers of ECL can use a middleware component called the ECL connector. Our connector hides the complexity of forming valid ECL messages and exposes the main protocol functions in the form of handlers that can be quickly implemented.

WWW 2004, May 17–22, 2004, New York, New York, USA.
ACM 1-58113-912-8/04/0005.

2. WHAT IS ECL?
The ECL protocol conforms to IMS DRI specifications[1]; it adds clarity to IMS DRI and provides definitions in areas where IMS DRI is not specific enough for the implementation of the protocol. The ECL defines actions that correspond with IMS DRI main functions: search/expose, submit/store, gather/expose and request/deliver. Along with documentation, XML schemas for each ECL request and response are defined. Developers can use these schemas to validate their implementation of ECL messaging protocol.

The ECL extends the IMS DRI protocol to include definitions of the GATHER service based on OAI harvesting protocol [2]. The only exception is that GATHER uses IMS/IEEE metadata schema while OAI uses Dublin Core schema.

Another gray area in the IMS DRI definitions is its recommendation of Xquery as the search query language. XQuery is a very powerful query language but the technology is rather new and many issues are still unresolved—notably the security, the differences in metadata schemas, and the backend support of XQuery. Consequently, most existing repositories are not ready to switch to XQuery databases. To deal with this problem, the ECL

used predetermined query patterns. These patterns can be parsed with any programming language and processed on any relational database.

To join the eduSource network, developers can download resources from our Web site (www.edusplash.net/technical/index.html). These provide all the necessary documentations for implementing ECL protocol as well as a Java ECL package. Java ECL is a plug-in Java API that facilitates the implementation and the deployment of ECL.

3. ECL MESSAGE ANATOMY

The ECL message has two parts: header and payload. The header contains information that allows an ECL enabled system to convey a message to its destination: communication id, sender information, request type, and login information. The payload is either the content of the request or the response. There are four types of requests (Search, Submit, Gather, and Request) and four types of responses: (SearchExpose, Store, GatherExpose, and Deliver).

4. IMPLEMENTATION

There are two distinct actors for the ECL implementation: the service provider and the client. A provider is typically a digital repository that provides ECL services. A client is an application accessing services provided by ECL-enabled repositories.

4.1 Service Provider

For repositories implemented on a Java platform, the implementation of ECL is very simple. These repositories should already have utilities to search and access their metadata databases. Using our ready-to-use Java ECL connector implementation, developers implement service handlers for the services they want to provide and deploy their ECL services. Java ECL distribution provides an Ant build file that automates the deployment of ECL onto Apache Axis SOAP engine.

For a repository running on another platform, the implementation is not much more difficult. Along with the documentation, developers can use Java ECL as a reference. The only major task is implementing the utilities for parsing and building ECL messages using XML. Our experience with Python tells us that an experienced Python programmer can implement ECL protocol from scratch in Python in a few days.

We are currently working with the LionShare group to develop security models for ECL that is flexible and easy to use. A service provider will be able choose a security model that is most suitable for its security policy and upload the model information into the UDDI registry, which will be made available to clients during ECL service discovery process. The clients use this information to select the appropriate security model and make necessary authentication to access the services.

4.2 Client

On the client application side the implementation is also straightforward. The major requirement is a basic knowledge of XML. ECL provides definition and schema for each request and response as well as the WSDL generated by most SOAP engines. Python, Perl, Java, and .NET have utilities for XML and making SOAP requests. ECL specifically avoided using SOAP's complex type to keep a high level of interoperability. The only input/output type requirements ECL poses for the transport protocol are *string* and *file attachment* that are supported by most standard SOAP libraries. Once the developers know how to parse and build ECL messages, the call to ECL services only requires a few lines of code, and the WSDL file obtained from ECL-enabled repositories are usually compatible across all platforms.

5. INTEROPERABILITY

ECL was designed for interoperability; rather than enforcing IEEE LOM and web services we elected to use an XML representation and messaging approach.

The **ECL connector** enables us to connect several repositories and tools to the eduSource network. In a three-day session with the developers from CAREO, Explora, Adlib, and Pond repositories we implemented full ranges of ECL services in connecting all of these repositories. In addition, we implemented an add-on interface into our respective end-user tools for communicating with those repositories (Splash, Explora, and Aloha). This meeting verified our assumptions about the ECL's ease in connecting existing repositories.

The second mechanism supporting interoperability is an **ECL Gateway** that provides a framework for bridging ECL with other protocols. Several ECL Gateways were instantiated to bridge the ECL network with SMETE, EdNa, and RDN by converting ECL to the corresponding protocols. Currently we are working on interconnecting ECL with the OKI-LionShare project [3, 4].

ECL also deals with the interoperability of the metadata. The protocol itself is agnostic to the metadata standard used. It provides guidelines for developers to convert metadata from one format to another. Currently ECL connector supports Dublin Core, IMS/IEEE LOM (XML and RDF bindings) and new XSLT conversion stylesheets can be introduced into ECL to support other metadata formats. Practically, ECL can be used on any repository.

6. CONCLUSIONS

ECL has achieved its interoperability objectives and shows a promising future. The protocol is flexible and adaptive. It allows repositories implemented on different platforms and using different metadata schemas to connect into a single network.

7. REFERENCES

[1] IMS DRI (2003). IMS Digital Repositories Interoperability [On-line]. Available at: http://www.imsglobal.org/digitalrepositories/driv1p0/imsdri_bestv1p0.html

[2] OAI. The Open archives Initiative Protocol. [On-line]. Available at: http://www.openarchives.org/OAI/

[3] Open Knowledge Initiative [On-line]. Available at: http://web.mit.edu/oki/

[4] LionShare P2P project. [On-line]. Available at: http://lionshare.its.psu.edu