The eduSource Communication LANGUAGE (ECL)

Version 0.2 – Working Document
Jean-Francois Arcand

DRAFT

Version	Authors	Change	Date
0.1	Jean-Francois Arcand	Create document	11/15/2002
0.2	Jean-Francois Arcand	Add XML Schema section Add Authentication section Re format the document	11/29/2002

Introduction

<<Goal of this working group>>

The ECL Protocol

In their transport form, messages are represented as xml expression. The first element of the message is a word which identifies the message-type being communicated, which defines the principal meaning of the message. There then follows a sequence of message attributes, introduced by XML attribute keywords. The body of the message contains the content of the message (named here protocol), encoded as an expression in some formalism (message can have their own XML schema formalism). Other attributes help the message transport service to deliver the message correctly (e.g. sender and receiver), help the receiver

Ongoing conversations between eduSource application often fall into typical patterns. In such cases, certain message sequences are expected, and, at any point in the conversation, other messages are expected to follow. These typical patterns of message exchange are called protocols. A designer of eduSource applications has the choice to make the eduSource application sufficiently aware of the meanings of the messages (by implementing the proper XML Schema). This, however, places a heavy burden of capability and complexity on the eduSource application implementation. An alternative, and very pragmatic, view is to pre-specify the protocols, so that a simpler eduSource application implementation can nevertheless engage in meaningful conversation with other eduSource application simply by carefully following the known protocol.

This section of the specification details such protocol, in order to facilitate the effective interoperation of simple and complex eduSource application. No claim is made that this is an exhaustive list of useful communication message, nor that they are necessary for any given application. The protocol is given pre-defined names: the requirement for adhering to the specification is:

"An eduSource application need not implement any of the standard protocols (XML schema), nor is it restricted from using other protocol names. However, if one of the standard protocol names is used, the eduSource application must behave consistently with the protocol specification given here."

Note that eduSource application can engage in multiple dialogues, perhaps with different eduSource application, simultaneously. The term conversation is used to denote any particular instance of such a dialogue. Thus, the eduSource application may be concurrently engaged in multiple conversations, with different eduSource application within different protocols. The remarks in this section which refer to the receipt of messages under the control of a given protocol refer only to a particular conversation.

ECL XML Schema

```
<schema xmlns='http://www.w3.org/2000/10/XMLSchema'
                    targetNamespace='http://edusource.surrey.sfu.ca/ecl'
                    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                    version="1.0">

  <xsd:element name='ecl_1_0'>
    <xsd:complexType>
      <xsd:sequence minOccurs='1' maxOccurs='1'>
        <xsd:element ref='message'/>
        <xsd:element ref='message-extension'/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name='message'>
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref='protocol' minOccurs='1' maxOccurs='1'/>
      </xsd:sequence>
      <xsd:attribute name='action' type='string' use='required'/>
      <xsd:attribute name='sender' type='string' use='required'/>
      <xsd:attribute name='receiver' type='string' use='required'/>
      <xsd:attribute name='in-reply-to' type='string' use='optional'/>
      <xsd:attribute name='reply-with' type='string' use='optional'/>
      <xsd:attribute name='conversion-id' type='ID' use='required'/>
    </xsd:complexType>
  </xsd:element>
```

[…]

Where attribute:

action

CANCEL: The action of canceling some previously requested (REQUEST) action which has temporal extent (i.e. is not instantaneous).
REQUEST: The action of telling another eduSource application that an action was attempted but the attempt failed. The sender always assumes that an ANSWER will come back.
INFORM: The sender informs the receiver of some action. The sender will not receive any confirmation from the receiver.
REFUSE: The action of refusing to perform a given action, and explaining the reason for the refusal.
SUBSCRIBE: The sender requests the receiver to perform some action subscribe: The

act of requesting a persistent membership to an eduSource application to notify the sender of the value of a message, and to notify again whenever the message identified by the reference changes. An INFORM message will be sender to every subscriber. **ANSWER:** The receiver performs an action and returns the result to the sender. This action is activated by a REQUEST action.

sender

Denotes the identity of the sender of the message, i.e. the name of the eduSource application.

receiver

Denotes the identity of the intended recipient of the message. Note that the recipient may be a single eduSource application name, or a tuple of application names (separated by a coma). This corresponds to the action of multicasting the message. Pragmatically, the semantics of this multicast is that the message is sent to each eduSource application named in the tuple, and that the sender intends each of them to be recipient of the content encoded in the message.

reply-with

Introduces an expression which will be used by the eduSource application responding to this message to identify the original message. Can be used to follow a conversation thread in a situation where multiple dialogues occur simultaneously. E.g. if eduSource application i sends to eduSource application j a message which contains reply-with query1, eduSource application j will respond with a message containing :in-reply-to query1.

in-reply-to

Denotes an expression that references an earlier action to which this message is a reply.

conversion-id

Introduces an expression which is used to identify an ongoing sequence of communicative message which together form a conversation. A conversation may be used by an eduSource application to manage its communication strategies and activities. In addition the conversation may provide additional context for the interpretation of the meaning of a message.

```
  <xsd:element name='protocol'>
    <xsd:complexType mixed='true'>
      <xsd:attribute name='namespace' type='string' use='required'/>
    </xsd:complexType>
  </xsd:element>
[…]
```

where attribute namespace define the protocol used (and XML Schema URI). An XML schema will be developed for every protocol the eduSource project will supports. This way eduSource project can support as many protocol as we defined.

```
  <xsd:complexType name="message-extension">
    <xsd:sequence>
      <xsd:element name="extension-element"
            minOccurs='0' maxOccurs='unbounded'/>
    </xsd:sequence>

    <xsd:attribute name="namespace"
          use="required"/>
    <xsd:attribute name="mustUnderstand"
          type="xsd:boolean"/>
  </xsd:complexType></schema>
```

The **message-extension** element is used by eduSource application that wants to add optional information to the message element. The **message-extension** is optional, but when the **mustUnderstand** element is equal to true, the receiver of the message needs to properly parse the **message-extension** element. If it cannot, then the message must be rejected.

Examples

(1) How we support OAI

```
<message type="REQUEST"  sender="Explor@"  receiver="Splash" conversion-id=1>
        <protocol namespace="http://www.eduSource.org/OAI_HTTP">
                <query=" http://ar.org/oai2?verb=GetRecord&identifier=
oai:ar:cs/0117&metadataPrefix=oai_dc"/>
</message>

<message type="ANSWER"  sender=" Splash"  receiver=" Explor@" conversion-id=2>
        <protocol namespace=" http://www.eduSource.org/OAI_HTTP" >
                // the query answer.
```

</message>

(2) How we support Z39.50

```
<message type="REQUEST"  sender="Explor@"  receiver="Splash" conversion-id=1>
        <protocol namespace="http://www.eduSource.org/Z39.50" >
                // do an Z39.50 query
</message>
```

Authentication <<Need to be discussed next meeting>>

SOAP does not define a standard way to authenticate users. HTTP standard way of authenticating user should be used until a standard emerge.

An eduSource application can authenticate a user/application to a web server using one of the following

- HTTP Basic Authentication
- HTTP Digest Authentication (http://www.ietf.org/rfc/rfc2617.txt)
- HTTPS Client Authentication

HTTP Basic Authentication, which is based on a username and password, is the authentication mechanism defined in the HTTP/1.0 specification. A web server requests a eduSource application to authenticate the user. As part of the request, the web server passes the realm (a string) in which the user is to be authenticated. The realm string of Basic Authentication does not have to reflect any particular security policy domain (confusingly also referred to as a realm). The eduSource application obtains the user name and the password from the user and transmits them to the eduSource application. The eduSource application then authenticates the user in the specified realm. Basic Authentication is not a secure authentication protocol. User passwords are sent in simple base64 encoding, and the target server is not authenticated. Additional protection can alleviate some of these concerns: a secure transport mechanism (HTTPS), or security at the network level (such as the IPSEC protocol or VPN strategies) is applied in some deployment scenarios.

Like HTTP Basic Authentication, HTTP Digest Authentication authenticates a user based on a username and a password. However the authentication is performed by transmitting the password in an encrypted form which is much more secure than the simple base64 encoding used by Basic Authentication, e.g. HTTPS Client Authentication. As Digest Authentication is not currently in widespread use, eduSource application containers are encouraged but not required to support it.

End user authentication using HTTPS (HTTP over SSL) is a strong authentication mechanism. This mechanism requires the user to possess a Public Key Certificate (PKC).

<<Need to be discussed>>
The authentication can occurs at the SOAP message level or using action Subscribe. I recommend we use it at the SOAP level using HTTP Digest Authentication