COLDEX

Collaborative Learning and Distributed Experimentation

Information Society Technologies Programme
Project number: IST-2001-32327

Functional Documentation

Tools Integration. UNED contribution

Deliverable Number:	D.7.2.2
Contractual Date of Delivery:	
Delivery Date:	24/01/05
Version:	1.0
Lead Partner:	
Contributing partners:	
Authors:	Verdejo, M.F.; Vélez, J.; Barros, B; Celorrio, C. Mayorga; J.I.
Contact:	ccelorrio@bec.uned.es

1 Introduction

Learning object repository provides a web user interface allowing COLDEX members to access its functionalities via a web browser. In spite of being a convenient way to get through to LOR contents, there is another important requirement. COLDEX partners' tools should be adequately integrated within the COLDEX system. Thus, these external tools are also clients that can access (exploit) the LOR.

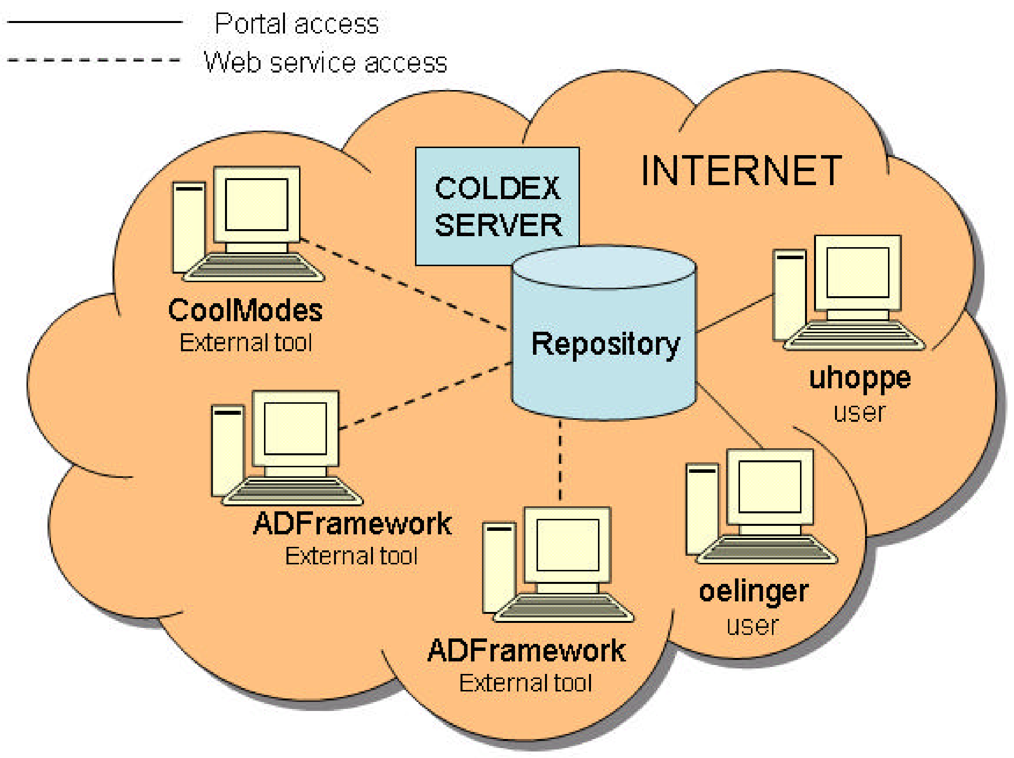

Figure 1. COLDEX access scenario

Figure 1 shows that tools need a protocol to interact with the LOR. To achieve this objective a web service based solution has been adopted. This deliverable document describes all web services implemented to meet this requirement.

2 LOR Web Services Exploitation

COLDEX web services permits having access to LOR functionalities from external tools. Following subsections describe the collection of implemented web services up to date. Each description includes several fields:

- Name. Representing the LOR web service name
- Description. A detailed explanation of service purposes and functionalities
- Input Parameters. Input parameters description
- Returning Output. Description of the contents and structure for the web service outcome

2.1 Login service

Name

loginLOR

Description

This web service permits accessing a COLDEX server. As a result of calling it, access is granted to all other web services. Hence, this is the first web service to be called in order to make use of the LOR from any external applications.

Input Parameters

String username. The name of the user trying to access LOR

String password. The password to authenticate this user

Returning Output

String successMessage. XML content with different possible values:

- Logged. When login is successfully done.
  ```
  <logged> username </logger>
  ```
- UserNotFound. When user is not found.
  ```
  <userNotFound/>
  ```
- BadPassword. When password does not match the user's
  ```
  <badPassword/>
  ```

2.2 Download Learning Object Service

Name

downloadLO

Description

This web service provides a mechanism for downloading an object from the LOR. Using this service requires specifying both repository and learning object IDs. The downloaded LO is delivered as a zip file including its content along with a XML metadata record (see deliverable D.7.2.2.).

Input Parameters

String repositoryId. The repository identifier.

String LOId. The Learning Object Identifier.

Returning Output

byte [] loData. The returning value contains the serialized binary data of the zip file containing the learning object and metadata record.

2.3 Upload learning Object Service

Name

uploadLO

Description

This web service permits uploading a learning object to the LOR. The learning object is sent as a byte stream representing its data. This data stream can either represent just a plain or a zipped file including content and XML based metadata record (see deliverable D.7.2.2). In addition, it is also necessary to specify the learning object file name and its type (such as 'Collide' or 'BeLife') and the ID repository where it is to be stored. ID.

Input Parameters

byte [] fileData. The stream of binary data representing contents.

String filename. The name of learning object file.

String type. The type of learning object (see Deliverable D.7.2.2)

String repositoryId. The repository identifier.

Returning Output

String successMessage. The returning value contains an XML-formatted message with possible different values and meanings:

- *Success*. When learning object has been correctly uploaded.

```
<success> message </success>
```

Error. When errors have been found during the uploading process.

```
<error> message </error>
```

- *userNotLogged.* When user is not logged in.

```
<userNotLogged/>
```

2.4 Searching Learning Objects service

Name

searchLOR

Description

This web service permits querying about LOR contents. In order to use this service, it is necessary to provide the target repository ID and a XML-formatted query statement (see input parameters for details). As a result of the ensuing searching process, an XML message is obtained. This message contains both a list of learning objects identifiers and the metadata record, which were requested by the query statement.

Input Parameters

String repositoryId. The repository identifier

String queryPattern. The query statement. A query pattern can contain two types of slots: slots with values, which are used as filters and slots without values, which represent requested attributes. Irrelevant attributes are not included. The next example shows such a pattern, which filters all learning object with 'Type' 'Collide' and also requests 'Name' and 'Palettes' metadata fields to be included in the result.

```
<query_pattern>
   <entity>
        <slot name="Type">
           <value>Collide</value>
       </slot>
       <slot name="Palettes"/>
       <slot name="Name" />
   </entity>
</query_pattern>
```

Returning Output

String queryResult. This query result consists of an XML message including several 'entity' labels. Each label represents a learning object result containing all metadata slots within the query pattern. These slots include metadata slot name and value (or value list if metadata is multi-valued).

```
<query_result>
   <entity id="4">
       <slot name="Name">
           <value>Maze</value>
       </slot>
       <slot name="Type">
           <value>Collide</value>
       </slot>
       <slot name="Palettes">
           <value>Maze Designer</value>
           <value>Moon</value>
           <value>Graphical </value>
           <value>RuleSet Discussion</value>
           <value>Discuss</value>
           <value>DrawPalette</value>
       </slot>
   </entity>
   <entity id="5">
       <slot name="Name">
           <value>Maze2</value>
       </slot>
       <slot name="Type">
           <value>Collide</value>
       </slot>
       <slot name="Palettes">
           <value>Maze Designer</value>
           <value>Moon</value>
           <value>Graphical Calculator </value>
           <value>RuleSet Discussion</value>
           <value>Discuss</value>
           <value>DrawPalette</value>
       </slot>
   </entity>
</query_result>
```

2.5 Get Public Repository Service

Name

getPublicRepository

Description

This web service can be used to obtain the LOR's public repository identifier. Recall that public repository is that that is visible for all COLDEX society.

Input Parameters

None

Returning Output

String repositoryList. An XML message containing both the public repository identifier and its name. For instance:

```
<repositories>
  <repository id="1">
      <name>Coldex Society (Public)</name>
  </repository>
</repositories>
```

2.6 Get Private Repository Service

Name

getPrivateRepository

Description

This web service can be used to obtain the private repository identifier of a user. The only parameter to be specified is the user name.

Input Parameters

String username. The user name.

Returning Output

String repositoryList. An XML message containing both the private repository identifier and its name. For instance:

```
<repositories>
  <repository id="28">
      <name>oelinger (User)</name>
  </repository>
</repositories>
```

2.7 Get Group Repository Service

Name

getGroupRepository

Description

This web service can be used to obtain the repository identifier of a group or community.

Input Parameters

String groupId. The group identifier.

Returning Output

String repositoryList. An XML message containing both the group repository identifier and its name. For instance:

```
<repositories>
  <repository id="7">
      <name>Maze (Group)</name>
  </repository>
</repositories>
```

2.8 Get User Groups Service

Name

getUserGroups

Description

This web service can be used to obtain all the identifiers of the groups a user belongs to.

Input Parameters

String userName. The user name.

Returning Output

String groupList. The list of groups the user belongs to. This list contains both group identifiers, names and types. Example:

```
<groups>
  <group id="2">
      <name>Coldex Society</name>
      <type>Society</type>
  </group>
  <group id="5">
      <name>Organizers</name>
      <type>Group</type>
  </group>
</groups>
```

2.9 Get Scenarios Service

Name

getScenarios

Description

This web service permits obtaining the list of scenarios, projects and activities stored in COLDEX server.

Input Parameters

None

Returning Output

String scenariosList. The list of scenarios, projects and activities stored in COLDEX server. Each scenario contains its name, description, topic, subtopic and the list of projects defined on it. Each project contains its name, description, challenge and the list of activities defined on it. Finally, each activity contains its name and description. Example:

```
<scenarios>
   <scenario>
    <name>Thinking about Biology</name>
    <description>This scenario ...</description>
    <topic>Biology</topic>
    <subtopic>Biodiversity</subtopic>
    <projects>
      <project>
        <name>Biology Photographs</name>
        <description>This project ...</description>
        <challenge>Study the plants ...</challenge>
        <activities>
           <activity>
              <name>Take Pictures</name>
              <description>...</description>
           </activity>
           ...
        <activities>
```

```
     </project>
    ...
    </projects>
  </scenario>
  ...
</scenarios>
```

3 WebServiceClient utility class.

WebServiceClient is a Java utility class that provides a simple and straightforward way to access LOR functionalities from any external tool. In order to achieve it, this class implements calls to previously described web services.

3.1 Requirements

In order to use WebServiceClient class it is necessary to install AXIS software. AXIS software is a framework for constructing SOAP processors such as clients, servers, gateways, etc. It also includes a simple stand-alone server, a server which plugs into servlet engines such as Tomcat, extensive support for the Web Service Description Language (WSDL), emitter tooling that generates Java classes from WSDL and a tool for monitoring TCP/IP packets.

3.2 Uploading learning object example

Following code listing shows an WebServiceClient class use example. This example deals with uploading an Active Document Learning Object to the public repository.

```
import org.apache.axis.client.Service;
import webServices.WebServicesClient;

public class ClientUploadLO
{
    public static void main(String [] args)
    {
            //Web Service URL
            String endpointURL = "...";
            String userLOR="admin";      // user name
            String passLOR="admin";      // password
            String repositoryID;         // repository id
            String loType="ADResultSet"; // learning object type
            String filePath="./resultXml_test.zip";  //file path

WebServicesClient wSC=new
WebServicesClient(endpointURL, userLOR, passLOR);

repositoryID = wSC.getPublicRepository();

String result =
wSC.uploadLO(filePath, loType, repositoryID);
            System.out.println("Upload result: " + result);
    }
}
```

4 References

- AXIS Web Services. http://ws.apache.org/axis/
- Deliverable D.7.2.2

5 Apendix A. WebServiceClient JavaDoc