OHF XDS SOAP Client

Architecture & API Documentation

Version 0.0.2

seknoop[AT]us[DOT]ibm[DOT]com | Sarah Knoop

Contents

1. Introduction

The Eclipse Foundation is a not-for-profit corporation formed to advance the creation, evolution, promotion, and support of the Eclipse Platform and to cultivate both an open source community and an ecosystem of complementary products, capabilities, and services. Eclipse is an open source community whose projects are focused on providing an extensible development platform and application frameworks for building software.

- www.eclipse.org

The Eclipse Open Healthcare Framework (EOHF) is a project within Eclipse formed for the purpose of expediting healthcare informatics technology. The project is composed of extensible frameworks and tools which emphasize the use of existing and emerging standards in order to encourage interoperable open source infrastructure, thereby lowering integration barriers.

- www.eclipse.org/ohf

The Integrating the Healthcare Enterprise (IHE) is an initiative by healthcare professionals and industry to improve the way computer systems in healthcare share information. IHE promotes the coordinated use of established standards such as DICOM and HL7 to address specific clinical needs in support of optimal patient care. Systems developed in accordance with IHE communicate with one another better, are easier to implement, and enable care providers to use information more effectively.

- www.ihe.net

The IHE Technical Frameworks are a resource for users, developers and implementers of healthcare imaging and information systems. They define specific implementations of established standards to achieve effective systems integration, facilitate appropriate sharing of medical information and support optimal patient care. They are expanded annually, after a period of public review, and maintained regularly by the IHE Technical Committees through the identification and correction of errata.

- http://www.ihe.net/Technical_Framework/index.cfm

This document describes the current release of the Eclipse OHF plugin implementation of the client side of the SOAP based transactions used in XDS; chiefly ITI-15: Provide and Register Document Set and ITI-16: Query Registry. We note that this plugin can potentially be used in other IHE SOAP based transactions; however, such use has not been discussed or implemented at this time. We also note that the term “client” is used to signify the entity producing and initiating a SOAP message exchange (request-response) with another entity.

2. Getting Started

2.1 Platform Requirements

Verify that the following platform requirements are installed on your workstation, and if not follow the links provided to download and install.

Eclipse SDK 3.2, or later	http://www.eclipse.org/downloads/
Java JDK 1.4.2, or later	http://java.sun.com/javase/downloads/index.jsp

2.2 Source Files

Information on how to access the Eclipse CVS technology repository is found on the eclipse wiki:

http://wiki.eclipse.org/index.php/CVS_Howto

Download from dev.eclipse.org/technology/org.eclipse.ohf/plugins

- org.eclipse.ohf.ihe.xds.soap

For details regarding plugin contents, see the README.txt located in the resources/doc folder of each plugin.

2.3 Dependencies

2.3.1 Other OHF Plugins

Plugin dependencies include the following from dev.eclipse.org/technology/org.eclipse.ohf/plugins

• org.apache.axis	Apache Axis 1.3 to support SOAP messaging
• org.apache.log4j	Debug, warning and error logging

2.3.2 External Sources

At this time, the XDS Soap Client is dependant on the Axis 1.3 API. This API does not provide an implementation of the javax.acitvation.DataHandler nor the javax.mail.internet.MimeMultipart packages needed for compilation and SOAP with attachment support. Java Mail 1.3.3 (mailapi.jar) and Java Activation Framework 1.0.2 (activation.jar) must be downloaded separately and incorporated into the build path or plugin dependencies list for this plugin. Take note of exact versions of these jars. These .jars can be found at

JAF v1.0.2:

http://java.sun.com/products/archive/javabeans/jaf102.html

JAVA MAIL v1.3.3:

http://java.sun.com/products/javamail/javamail-1_3_3.htmlResources

Additionally, we are experiencing problems with the Axis 1.3 support for SOAP Attachments (for explicit details see the following posting: http://mail-archives.apache.org/mod_mbox/ws-axisuser/200607.mbox/%3cb50b8b100607191155s4c7159e7i20f5c3b8bf4434c8@mail.gmail.com%3e). We have yet to hear a response to this posting, and for the time being the following steps are needed to enable attachment support (needed when using the XDS Document Source).

1. Download Axis 1.3 source code from: http://www.apache.org/dyn/closer.cgi/ws/axis/1_3
2. Unzip the content into a directory of your choosing.
3. In org/apache/axis/attachments/AttachmentsImpl.java comment out line 229 and lines 586 through 595 (Specifically, the operation being removed is "multipart = null;").
4. Re-compile this class and overwrite the existing class file in the axis.jar included in the org.apache.axis plugin for OHF.

2.4 Resources

2.4.1 Apache Axis 1.3

Further documentation and source code for Apache Axis 1.3 can be found at:

http://ws.apache.org/axis/

2.4.2 IHE ITI Technical Framework

Nine IHE IT Infrastructure Integration Profiles are specified as Final Text in the Version 2.0 ITI Technical Framework: Cross-Enterprise Document Sharing (XDS), Patient Identifier Cross-Referencing (PIX), Patient Demographics Query (PDQ), Audit trail and Node Authentication (ATNA), Consistent Time (CT), Enterprise User Authentication (EUA), Retrieve Information for Display (RID), Patient Synchronized Applications (PSA), and Personnel White Pages (PWP).

The IHE ITI Technical Framework can be found on the following website: http://www.ihe.net/Technical_Framework/index.cfm#IT.

Key sections relevant to the XDS Soap Client include (but are not limited to):

- Volume 1, Section 10
- Volume 2, Section 1, Section 2, Section 3.15, Section 3.16

2.4.3 Newsgroup

Any unanswered technical questions may be posted to Eclipse OHF newsgroup. The newsgroup is located at news://news.eclipse.org/eclipse.technology.ohf.

You can request a password at: http://www.eclipse.org/newsgroups/main.html**.**

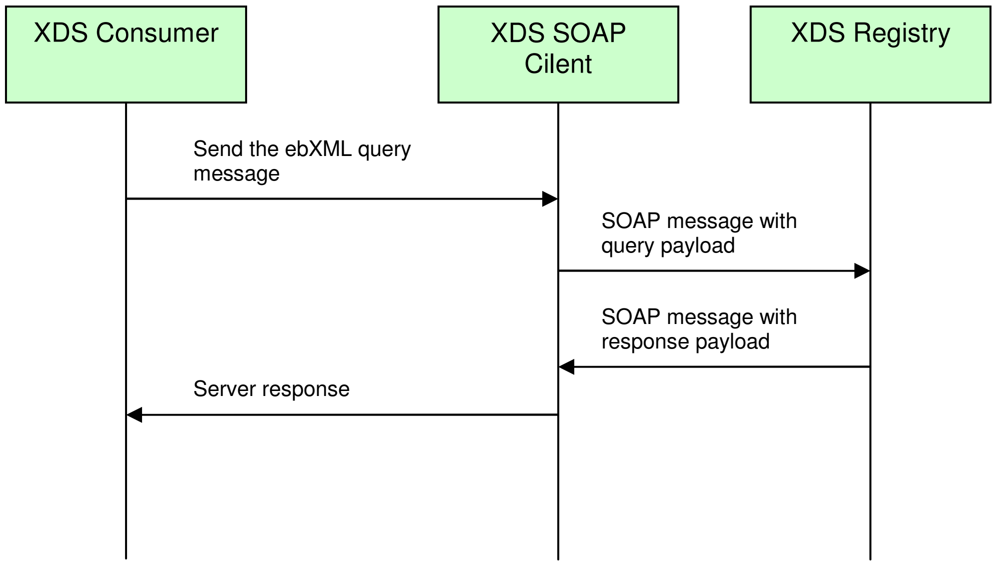
3. API Documentation

The OHF XDS SOAP Client is intended to provide a simple and common interface for the XDS client actors (Document Source and Document Consumer) to execute their respective transactions using SOAP. The SOAP Client constructs and SOAP message, adds any attachments and sends the message. Upon receiving a response from the server, the SOAP client un-wraps the SOAP Envelope from the response, returning to the XDS client only the contents of the SOAP Body as a DOM Element. We note that the API for this plugin is not stable due to the pending integration with other OHF components.

3.1 Use Case 1 – Query Registry

The OHF XDS SOAP Client constructs the SOAP message for the XDS Consumer Query Registry Transaction (ITI-16).

3.1.1 Flow of Execution


XDS Consumer
XDS SOAP Cilent
XDS Registry
Send the ebXML query message
SOAP message with query payload
SOAP message with response payload
Server response

3.1.2 API Highlights

The most significant method invoked in the above control flow is XDSSoapClient.send(). This interface method is described below in greater detail. The class XDSSoapClientAXIS, provides an implementation of this method using the Axis 1.3 API. The complete javadoc can be downloaded from the Eclipse CVS technology repository. See 2.2 for details.

XDSSoapClient.send()

`org.w3c.dom.Element`	**`send`**`(java.lang.String request, SoapAttachment[] attachments, java.lang.String url)` Constructs the SOAPMessage from the parameters, sends the messages to the designated url and processes the response SOAPMessage. **Parameters:** `request` - is assumed to be a serialized XML document string to be set as the contents of the SOAPBody. For the Source this is the SubmitObjectRequest element used for metadata. For the Consumer this is the AdhocQueryRequest element. This parameter must not be null, an Exception should be thrown if this is the case. `attachments` - an array of `SoapAttachment` objects. These are to be rendered as AttatchmentParts to the SOAPMessage. For the Document Source, these will be the documents submitted. The Consumer query transaction does not use attachments. In this case, the parameter can be null. `url` - the end destination of the SOAPMessage. This entity is assumed to have a SOAP interface and is able to generate a SOAPMessage in response. **Returns:** It is assumed, by XDS, that the SOAPMessage recieved in response will be contained as part of the SOAPBody. The contents of the SOAPBody are returned to the Source or Consumer as a DOM Element for further processing. **Throws:** `java.lang.Exception`

3.1.3 Sample Code

3.1.3.1 Description

The following sample code illustrates how the XDS Consumer queries the XDS Registry using the XDSSoapClient. The full context of this sample code can be obtained by downloading org.eclipse.ohf.ihe.xds.consumer/Consumer.java from the Eclipse CVS technology repository. See 2.2 for details.

3.1.3.2 Code

```
////////////////////////////////////////////////////////////////////////////////
//Code from Consumer.query():
//The Consumer is given a Query object, from which it forms a raw SQL query
//string. This SQL query string needs to be encased in an ebXML
//AdhocQueryRequest. The following call does this and returns the resultant XML
```

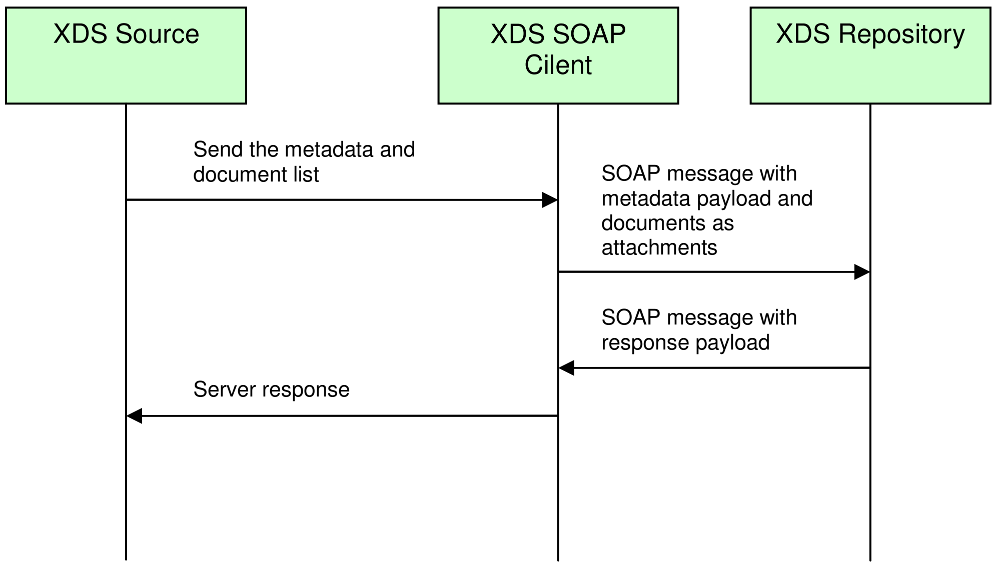
```
//as a string.
/////////////////////////////////////////////////////////////////////////////

String ebXMLQuery = null;
try {
      ebXMLQuery = formatEBXMLQueryString(sqlQueryString,returnType);
} catch (IOException e) {
      logger.fatal("Could not format ebXML for the query string.", e);
      throw e;
}
////////////////////////////////////////////////////////////////////////////////
//Now call the sendQuery() method to send the ebXML query to the registry.
//Sending the query is surrounded by auditing statements in order to capture
//this auditable event and its outcome in the audit record repository. Upon
//successful completion of the SOAP message exchange, the registry response is
//processed by the convert() method.
////////////////////////////////////////////////////////////////////////////////
int eventOutcome = ATNAAuditClient.SUCCESS_EVENT_OUTCOME;
AdhocQueryResponseType qr = null;
try {
      qr = sendQuery(ebXMLQuery);
} catch (Exception e) {
      eventOutcome = ATNAAuditClient.SERIOUS_FAILURE_EVENT_OUTCOME;
      throw e;
} finally {
      if (isDoAudit()) {
            auditor.audit(eventOutcome,initiatingUser, ebXMLQuery);
      }
}
return convert(qr);
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
//Code from Consumer.sendQuery():
//This is where the XDS Consumer invokes the XDSSoapClient (this.sender) to send
//the query. This support method is called by Consumer.query(), portions of
//which are above.
////////////////////////////////////////////////////////////////////////////////
Element reply = this.sender.send(ebXMLQuery, null, regURL);
if (reply == null) {
      throw new ConsumerException("No data returned from registry.");
}
```

```
return loadFromStream(new ByteArrayInputStream(reply.toString().getBytes()));
```

3.2 Use Case 2 – Provide and Register Document Set

The OHF XDS SOAP Client constructs the SOAP message for the XDS Document Source Provide and Register Document Set Transaction (ITI-15).

3.2.1 Flow of Execution


XDS Source
XDS SOAP Cilent
XDS Repository
Send the metadata and document list
SOAP message with metadata payload and documents as attachments
SOAP message with response payload
Server response

3.2.2 API Highlights

Here, we highlight the simplified SoapAttachment object used by the XDSSoapClient. The SoapAttachement object is used by the XDS Document Source to attach the clinical documents it wants to share to the SOAP message being sent to the XDS Repository. The complete javadoc can be downloaded from the Eclipse CVS technology repository. See 2.2 for details.

SoapAttachment()

```
SoapAttachment(java.lang.String data, java.lang.String mimeType,
java.lang.String contentId)
```

Constructs a SoapAttachment object given the following parameters

Parameters:

`data` - represents the attachment payload data (ie. contents)
`mimeType` - is the mime type of the attachment payload data
`contentId` - is the content id of the attachment payload data

3.2.3 Sample Code

3.2.3.1 Description

The following sample code illustrates how the XDS Source submits documents and corresponding XDS Metadata to the XDS Repository using the XDSSoapClient. The full context of this sample code can be obtained by downloading org.eclipse.ohf.ihe.xds.source/Source.java from the Eclipse CVS technology repository. See 2.2 for details.

3.2.3.2 Code

```
//////////////////////////////////////////////////////////////////////////////
//Code from Source.submit ():
//The XDS Source is provided metadata for the submission in the form of a
//ProvideAndRegisterDocumentSetType object. The XDS Source reformats the
//metadata into ebXML (SubmitObjectRequestType) using the "setTransformer"
//instance. The ebXML metadata is then serialized into a string by the
//"unpackSubmitObjectsRequest() method.
//////////////////////////////////////////////////////////////////////////////
EbXML_2_1ProvideAndRegisterDocumentSetTransformer setTransformer = new
EbXML_2_1ProvideAndRegisterDocumentSetTransformer();

ProvideAndRegisterDocumentSetType meta = txnData.getMetadata();

setTransformer.transform(meta);

String ebXMLMetadataString =
unpackSubmitObjectsRequest(setTransformer.getSubmitReq());


//////////////////////////////////////////////////////////////////////////////
//We now create a SoapAttachment for each document to be submitted
//////////////////////////////////////////////////////////////////////////////
SoapAttachment[] attachments = new SoapAttachment[txnData.getDocList().size()];

Iterator i = txnData.getDocList().iterator();

int count = 0;

while(i.hasNext()){

	Document docI = (Document)i.next();

	attachments[count] = new SoapAttachment(docI.getDocumentData(),

		docI.getDescriptor().getMimeType(), docI.getDocumentEntryUUID());

	count++;

}
```

```
/////////////////////////////////////////////////////////////////////////////////
//This is where the XDS Source invokes the XDSSoapClient (this.sender) to submit
//the documents and related XDS metadata. Sending the documents and metadata is
//surrounded by auditing statements in order to capture this auditable event and
//its outcome in the audit record repository. Upon successful completion of the
//SOAP message exchange, the registry response is processed by the
//processResponse() method.
/////////////////////////////////////////////////////////////////////////////////
int eventOutcome = ATNAAuditClient.SUCCESS_EVENT_OUTCOME;

Element reply = null;

try {

      logger.info("Submitting Provide and Register Document Set transaction to”
      + “Repository: " + repositoryURL);

      reply = this.sender.send(ebXMLMetadataString, attachments,
      this.repositoryURL);

} catch (Exception e) {

      eventOutcome = ATNAAuditClient.SERIOUS_FAILURE_EVENT_OUTCOME;

      logger.error("Error sending SOAP message to the repository. Error was " +
      e.getMessage(), e);

      if (isDoAudit()) {

            auditor.audit(eventOutcome, initiatingUser,
            this.formatPHIData(txnData));

      }

}

processResponse(reply, txnData, initiatingUser);
```

4. Security

4.1 Node Authentication

Node Authentication is pending integration with other OHF components providing this functionality. Currently, SOAP transactions are sent over an insecure connection. We are aware of this issue and are working on an alternative method to support mutual node authentication while integration is pending.

4.2 Auditing

Auditing to an IHE Audit Repository is not done by this plugin. The act of auditing a transaction is to take place on the implementation layer that uses this plugin for its SOAP communications. In other words, IHE actors (such as the XDS Document Source and XDS Document Consumer) using this plugin are expected surround calls to this API with appropriate auditing measures.

5. Configuration

Logging is the only configurable aspect of this plugin at this time. This most likely will change when integration with other OHF plugins is complete. For more information about logging, consult Section 6 of this document. The following is a list of anticipated configurable aspects:

- Java keystore file for SSL communication
- Java keystore pass for SSL communication
- Java truststore file for SSL communication
- Java truststore pass for SSL communication

6. Debugging Recommendations

The XDS Soap Client uses Apache Log4j. If you are experiencing bugs related to SOAP, you may enable debug level logging by adding the following category to your log4j XML configuration file.

```
<category name="org.eclipse.ohf.ihe.xds.soap">
   <priority value="debug" />
</category >
```

For more information about Log4j please see: http://logging.apache.org/log4j/docs/

For an example log4j XML configuration file and to see how it is loaded at run time see `org.eclipse.ohf.ihe.xds.source/resources/conf/submitTest_log4j.xml` and `org.eclipse.ohf.ihe.xds.source/src_tests/SubmitTest.java`, respectively. These can be obtained by downloading the plugin `org.eclipse.ohf.ihe.xds.source` from the Eclipse CVS technology repository. See 2.2 for details.

7. Pending Integration and API changes

Below is a laundry list of anticipated changes to the XDS SOAP Client

1. Integration of Node Authentication – It is not clear at this point how this will affect the current API of the XDS SOAP client. We are working with the other OHF members supplying the functionality to ensure that changes are minimized.

8. Additional Sections – repeat as necessary

Any additional sections needed are added at this point.

9. Glossary

Define any non-common knowledge terms or acronyms here. Provide web-site reference if applicable.