OHF ATNA Audit Client

Architecture & API Documentation

Version 0.0.2

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

Contents

1. Introduction 3
2. Getting Started 4
   - 2.1 Platform Requirements 4
   - 2.2 Source Files 4
   - 2.3 Dependencies 4
     - 2.3.1 Other OHF Plugins 4
     - 2.3.2 External Sources 4
   - 2.4 Resources 4
     - 2.4.1 BSD and Reliable Syslog 4
     - 2.4.2 IHE ITI Technical Framework 4
     - 2.4.3 Newsgroup 5
3. API Documentation 6
   - 3.1 Use Case 1 – Query Registry PHI Import Event 6
     - 3.1.1 Flow of Execution 6
     - 3.1.2 API Highlights 6
     - 3.1.3 Sample Code 7
       - 3.1.3.1 Description 7
       - 3.1.3.2 Code 7
4. Security 9
   - 4.1 Node Authentication 9
5. Configuration 10
6. Debugging Recommendations 11
7. Pending Integration and API changes 12
8. Additional Sections – repeat as necessary 13
9. Glossary 14

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 IHE ITI Technical Framework Transaction ITI-20: Record Audit Event for use by any IHE Actor or healthcare application.

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.atna.audit

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

2.3 Dependencies

This plugin has very few dependencies at this time.

2.3.1 Other OHF Plugins

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

- org.apace.log4j Debug, warning and error logging

2.3.2 External Sources

This plugin has no external dependencies, yet.

2.4 Resources

2.4.1 BSD and Reliable Syslog

Eventually when we fill out the auditing client with a real implementation, we'll have to say something here.

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 OHF ATNA Audit Client include (but are not limited to):

- Volume 1, Section 9 and Appendices A,B and G
- Volume 2, Section 1, Section 2, Section 3.20 and Appendix K

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.

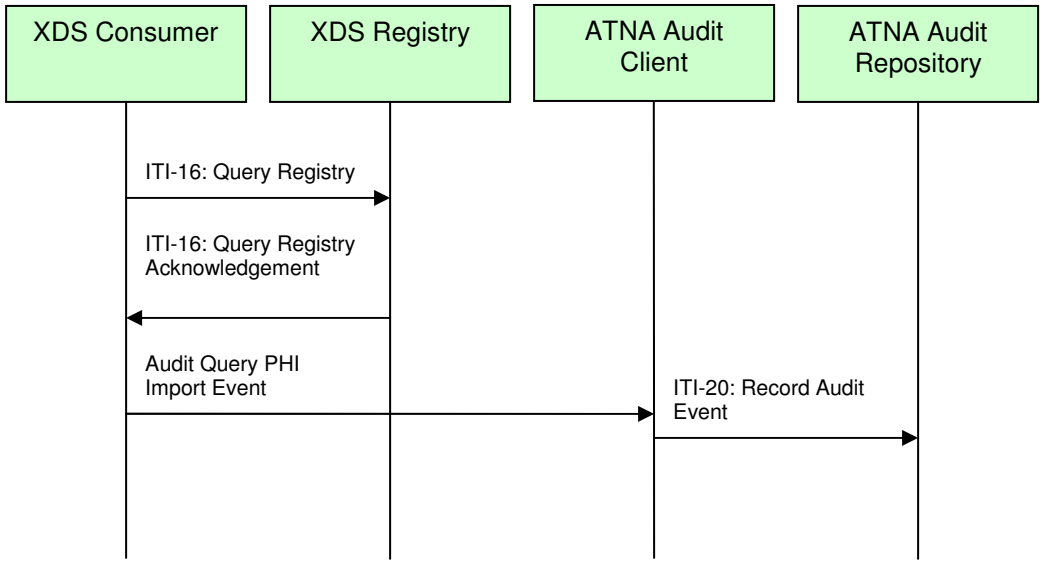
3. API Documentation

The ATNA Audit Client is intended to provide a simple and common interface for any IHE actor to participate in the ITI-20: Record Audit Event transaction. It constructs and sends the audit log message to the audit repository when an event occurs. 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 PHI Import Event

This audit event occurs when the XDS Document Consumer issues a query for documents to the XDS Registry and subsequently records the fact that it has received patient information.

3.1.1 Flow of Execution



.

3.1.2 API Highlights

The most significant method invoked in the above control flow is ATNAAuditClient.audit(). This interface method is described below in greater detail. The ATNAAuditClientDummy class, provides an initial, "place-holding", implementation of this method. The complete javadoc can be downloaded from the Eclipse CVS technology repository. See 2.2 for details.

ATNAAuditClient.audit()	
void	**audit**(int eventOutcome, java.lang.String initiatingUser, java.lang.String transactionPayload)

	Client side interface for transaction ITI-20: Record Audit Event Constructs the audit message and sends it to the audit repository on behalf of an IHE actor. **Parameters:** `eventOutcome` - status code of the event to be audited (success or failure to access patient(s) information) `initiatingUser` - the id of the entity that triggered the transaction concering the patient(s) information, thus triggering the audit event `transactionPayload` - transactional data used to obtain information about the patient(s) or containing information about the patient(s)

3.1.3 Sample Code

3.1.3.1 Description

The following sample code illustrates how the XDS Consumer accomplishes auditing using the ATNAAuditClient. 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.sendQuery():
//Consumer is about to send the ebXML formatted 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
//completion of the SOAP message exchange, the outcome (success, failure, etc.)
//of the query, along with PHI data and initiating user information are sent to
//the audit repository.
////////////////////////////////////////////////////////////////////////////


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);
```

```
        }
}
```

4. Security

4.1 Node Authentication

Node Authentication is pending integration with other OHF components providing this functionality. Currently audit messages are logged to a local log, rather than being sent out to an audit repository. This functionality is pending integration with other OHF components as well.

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
- Audit Record Repository url
- Audit Record Repository port

6. Debugging Recommendations

The ATNA Audit client uses Apache Log4j. If you are experiencing bugs related to auditing, you may enable debug level logging by adding the following category to your log4j XML configuration file.

```
<category name="org.eclipse.ohf.ihe.atna.audit">
   <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 ATNA Audit Client

1. Integration of Node Authentication and Auditing – It is not clear at this point how this will affect the current API of this plugin. We are working with the other OHF members supplying the functionality to ensure that changes are minimized … but most likely this API will endure a fair amount of revision.

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.