Open Healthcare Framework Bridge

Architecture & API Documentation

mattadav@us.ibm.com | Matthew Davis
kxjiang@us.ibm.com | Kelvin Jiang
monvura@us.ibm.com | Melih Onvural
ioprenc@us.ibm.com | Ivan Oprencak

Contents

1. Introduction

The United States Government is committed to apply modern information technology to private clinics and public hospitals nationwide by 2014. In 2006, President George W. Bush listed five key policies to modernize healthcare technologies, one of which was to implement “a nationwide information network [that] will protect the privacy of a patient's medical information while making health information available in real-time.”

The Eclipse Foundation is dedicated to design a common healthcare framework that will interconnect clinics and hospitals both on a regional level and on a national level to heed the call to build an integrated healthcare infrastructure. The first step is to provide existing EMR management systems with a platform-independent plug-in that defines patient information search and retrieval methods to and from an EMR repository. This documentation will address the design and implementation of this plug-in.

2. Technology Documentation

This chapter describes the sample PHP application that demonstrates the OHF Bridge technology.

2.1 OHF Bridge Demo

The following demo is for the initial standalone version of OHF Bridge, utilizing the OHF PIX, PDQ and XDS plug-ins. Some variation may be made in the steps of the demo; however, due to the limited nature of the repository data, irrelevant or bad data may be returned. The steps outlined returned relevant, working data at edit time.

2.1.1 File Locations

The Bridge demo currently resides at

http://ibmod235.dal-ebis.ihost.com:8080/ohf/demo/index.php,

The WSDL files currently reside at:

http://ibmod235.dal-ebis.ihost.com:8090/bridge/services/ohf-pdq-bridge?wsdl (Search patients)

http://ibmod235.dal-ebis.ihost.com:8090/bridge/services/ohf-xds-bridge?wsdl (Retrieve documents)

2.1.2 Demo Steps

Demonstrate Patient Search Functionality

1. Start by going to the PDQ Search Page at the link above
2. Enter as many or as few search terms as you would like. Recommended search (i.e. known results) come from searching for the state Wisconsin. **Select "Wisconsin" in the State drop-down**.
3. **Click "Search for Patients"**

   This will return a list of all patients that match your search terms in the PIX/PDQ repository. The first name on the list should be "Joyce Murphy".

Demonstrate Document Retrieval Functionality

1. To search for documents in the XDS Registry, **click "Search Documents" under the entry for "Joyce Murphy"**.

   Now a list of documents match Joyce Murphy's Patient ID, from the XDS registry, will show up. Some of these documents do not work, due to bad data in the repository or other problems. However, the "Patient Generated Medical History" documents will work.

2. To view a document, **click "View Document" next to any of the "Patient Generated Medical History" documents**.
3. An XML file should appear with relevant information. Any file with a matching MIME type should also work! At this time, only the XML files work.

This completes the basic PIX/PDQ/XDS demonstration using OHF Bridge in PHP. Variations may be made in the above steps (i.e. you can search by names, other states, etc) but, due to the limited nature of the repository, valid data may not be available.

2.1 Newsgroup Help

Several developers are currently working on OHF Bridge, and they are happy to help others to implement this technology in your EMR application. If you have any question about the implementation of OHF Bridge, please post it on the Eclipse OHF newsgroup.

The newsgroups is located at **news://news.eclipse.org/eclipse.technology.ohf**

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

3. Sample Code

The OHF Bridge provides a very simple API for patient search and document retrieval. This chapter describes the “hello world” code to search for a patient and to retrieve a document.

3.1 Search Patients

```
<?php
/**
 * Simple demonstration of searching for PDQ Records
 * in the registry
 */

require_once dirname(__FILE__) . '/conf.php';
require_once dirname(__FILE__) . '/OHFSoapClient.php';
require_once dirname(__FILE__) . '/PDQSearchObject.php';

// If a search term was submitted, do search.
if ($_POST) {
    $SOAPClient = new OHFSoapClient();

    $searchObj = new PDQSearchObject();
    $searchObj->setFields($_POST);

    // Retrieving the PDQ Records
    $results = $SOAPClient->searchPatients($searchObj);
    if (PEAR::isError($results)) {
        // If we received a PEAR Exception, then print debug information
and exit.
        echo "Error - Unable to query database: ". $results-
>getMessage();

        if (OHF_DEBUG) {
            echo '<pre>';
            print_r($results->getBacktrace());
            echo '</pre>';
        }
        exit();
    }
}
?>
```

3.2 Retrieve Documents

```
<?php
/**
 * Simple demonstration of searching for XDS documents
 * in the registry
 */

require_once dirname(__FILE__) . '/conf.php';
require_once dirname(__FILE__) . '/OHFSoapClient.php';

// If a search term was submitted, do search.
if ($_REQUEST['patientId']) {
    $SOAPClient = new OHFSoapClient();

    // Retrieving the XDS Documents
    $results = $SOAPClient-
>searchDocumentsByPatientId($_REQUEST['patientId']);
    if (PEAR::isError($results)) {
        // If we received a PEAR Exception, then print debug information
and exit.
        echo "Error - Unable to query database: ". $results-
>getMessage();

        if (OHF_DEBUG) {
            echo '<pre>';
            print_r($results->getBacktrace());
            echo '</pre>';
        }
        exit();
    }
}
?>
```

4. API Documentation

The following web services are implemented in Java using Apache AXIS and are callable via a general web service consumer. An automatically generated WSDL file is provided as a descriptor for these web services and can be used directly for consumption purposes.

Note: The following API is experimental and is subject to change, including web service association, operation names, prototype definitions and data structures. The following will be updated periodically to reflect changes in the general web service API.

4.1 Search Patients

For searching patients, the PDQ Bridge web service will take the fields from a search of patient demographic information, and return the results of a query on a PIX/PDQ Repository. The results of the search should allow users of an EMR system to find the documents associated with the individual(s) that are returned based on the values of the search fields.

4.1.1 Methods

SearchPatient

RPC call that sends out a local object that is mapped to a PDQQueryObject, and populated with values from the search demographic search form. Returns the result of the query as an array of type PDQQueryObject.

Request

Parameter	Type	Description	Required
q	PDQQueryObject	Object of Strings which are the demographic fields upon which one could search for patients	Yes

Response

searchPatient returns an array of type PDQQueryObject which is the result of the query run against the information collected in the search form.

4.1.2 Object Types

PDQQueryObject

Simple object composed of the fields necessary to query against patient information during the SOAP transit

Fields

Name	Type	Description	Required
patientId	String	The patient id by which an individual is stored in the repository	No
lastName	String	A patient’s last name	No
firstName	String	A patient’s first name	No
middleName	String	A patient’s middle name	No
nameSuffix	String	A patient’s suffix in their name (such as Jr., Sr., III, etc.)	No
namePrefix	String	A patient’s prefix to their name (such as Mrs., Mr., Dr., etc.)	No
nameTitle	String		No
sex	String	The patient’s gender	No
dateOfBirth	String	A patient’s date of birth	No
addressStreet	String	A patient’s street address	No
addressCity	String	A patient’s city address	No
addressCountyOrParish	String	A patient’s county or parish based on the local region in which they live and its terminology	No
addressStateOrProvince	String	A patient’s state or province based on the local region in which they live and its terminology	No
addressCountry	String	A patient’s nation of residence	No
addressZipOrPostalCode	String	A patient’s zip code or postal code based on the local region in which they live and its terminology	No
addressType	String	A descriptor for what address the patient is providing (business, home, P.O. Box, etc.)	No
addressOtherDesignation	String		No
phoneHome	String	A patient’s home telephone number	No
phoneBusiness	String	A patient’s work telephone number	No

4.2 Retrieve Documents

For retrieving documents, the XDS Bridge web service provides the necessary functionality to interface with the XDS registry and repository for searching, retrieving and submitting EMR documents. The XDS Bridge provides native functionality for searching the XDS registry for a list of records based on a unique patient identifier. This identifier may either be used directly or referenced indirectly via a patient search request.

4.2.1 Methods

SearchByPatientId

RPC call that searches the XDS registry for a list of documents associated with a given patient ID. Returns an array of objects of type XDSDocument that contain document metadata and information for further retrieval.

Request

Parameter	Type	Description	Required
patientId	String	Unique patient identifier for document searches.	Yes

Response

SearchByPatientId returns an array of objects of type XDSDocument or null if no responses found.

Exceptions

SearchByPatientId throws SOAP exceptions, to be handled by the consumer, on the event of server errors.

GetDocumentByUniqueId

Queries for and retrieves an XDS document from the repository based on the “unique identifier” assigned to each document placed in the repository. Returns a single object of type XDSDocument that contains document data and metadata.

Request

Parameter	Type	Description	Required
uniqueId	String	“Unique ID” for which to search/retrieve document from the registry for.	Yes

Response

Returns a single object of type XDSDocument with the document contents and metadata or null if the document was not found.

Exceptions

getDocumentByUniqueId throws SOAP exceptions, to be handled by the consumer, on the event of server errors.

GetDocumentByUUID

Queries for and retrieves an XDS document from the repository based on the Universally Unique Identifier (UUID) assigned to each document placed in the registry and repository. Returns a single object of type XDSDocument that contains document data and metadata.

Request

Parameter	Type	Description	Required
uuid	String	Universally unique identifier (UUID) for which to search/retrieve document from the registry for.	Yes

Response

Returns a single object of type XDSDocument with the document contents and metadata or null if the document was not found.

Exceptions

getDocumentByUniqueId throws SOAP exceptions, to be handled by the consumer, on the event of server errors.

4.2.2 Objects Types

XDSDocument

Simple data object for holding and mapping XDSDocument data while in transit via a SOAP envelope.

Fields

Name	Type	Description	Required
document	String	Base64-encoded binary or ASCII contents for the document.	No
documentTitle	String	Title of the document	No
message (*deprecated*)	String	Server message for error handling.	No
mimeType	String	MIME type for the document in transit.	No
size	Integer	Size in bytes of the document in transit	No
status (deprecated)	Integer	Server status code for document.	No
uuid	String	Universally unique identifier (UUID) for the document in transit.	Yes
uniqueId	String	"Unique ID" for document in transit.	Yes

5. Glossary

Eclipse:

Eclipse is an open source framework and platform for building software. The OHF Bridge uses the latest server-side OSGi technologies that will be included in the upcoming release of Eclipse 3.2. Specifically, OHF Bridge utilizes the server-side Equinox/OSGi technology, which is currently an incubator project within Eclipse. For more information, please go to **http://www.eclipse.org/equinox/**.

Electronic Medical Records (EMR):

EMRs are the patient medical records in electronic formats. The EMR data accepted by OHF Bridge fully conform to the HL7 international healthcare standard.

Interoperability Stack:

The Healthcare Repository is a physical datacenter that hosts patient medical documents. It has two separate components: the registry that hosts the metadata for patient and document search, and the repository that stores the contents of medical documents. In the event of patient search, the search is done in the registry; a list of matching patient is then returned. In the event of document retrieval, the registry accesses the repository, pulls the document and returns it. If the EMR application requests a creation, change or deletion of a document, the repository will first take the appropriate action and update the document, and then it will request a change in the metadata within the registry. A security component lies on top of both the repository and the registry and performs the permission handling.

Open Healthcare Framework (OHF):

OHF leverages the extensibility of Eclipse to create a set of standards and technologies to connect clinics and hospitals nationwide. For more information, please go to **http://www.eclipse.org/ohf/.**

OHF Bridge:

OHF Bridge is the first step towards implementing the Open Healthcare Framework. It is an open source, platform independent plug-in that can be implemented into existing EMR applications to add the capability of centralizing and searching of patient data. For more information, please go to **http://wiki.eclipse.org/index.php/OHF#OHF_in_Action**.

OHF Plug-ins:

The OHF Plug-ins includes three distinct mechanisms (PDQ, PIX, and XDS), each consists of two components: the component that retrieves files (the consumer component) and another component that creates files (the source component).

The PDQ plug-in handles the search of patient with demographic information including patient ID, full name, address, gender, date of birth, and contact information and returns a list of possible matches. Next, the PIX plug-in handle the cross-reference and comparison of patient metadata. Lastly, the XDS plug-in finds a list of documents for the matching patient and retrieves the full content of a selected document.