AMERICAN IMMUNIZATION REGISTRY ASSOCIATION

Immunization Information Systems for a New Era

Evaluation of Data Exchange Technologies for Communicating with IIS

For Immunization Information Systems (IIS) to be truly interoperable with other information systems, such as other IIS or electronic health records (EHRs), three layers—the application layer, data exchange layer and messaging layer—must be in place and use well-defined and accepted standards. For several years, public health entities have been promoting the use of the HL7 messaging standard by developing implementation guides for a variety of public health information systems, including IIS. These guides explain how to implement the standard HL7 message format in an IIS so that the IIS can send and/or receive messages in the HL7 message format.

Beyond the message layer is the data exchange layer—the layer that enables IIS to physically transmit the data contained in the HL7 message to a receiving information system or for a sending information system to transmit it to the IIS. To understand the role of the data exchange layer, it may be useful to provide an analogy of an apple truck collecting apples from an apple orchard and dropping them off at a grocery store. The apple orchard is the originating information system, apples are the data, the crates in which the apples are stored are the message format, the truck that carries the apples in their crates to the store is the data exchange layer, and the grocery store is the receiving information system.

This guide is a first step in determining which of the currently used data exchange layer methods should be promoted for use with IIS, much as existing messaging formats were reviewed and HL7 was identified as the messaging format most likely to meet the needs of public health systems.

Methods Examined

Many different methods of transmitting the data are currently available and in use, ranging from sending data in emails, saving to DVD, and sending via HTTP or SOAP. The Transport Layer Group of the American Immunization Registry Association (AIRA) originally intended to review, in detail, the following seven methods:

- Email
- Hypertext Transfer Protocol (HTTP)
- Simple Object Access Protocol (SOAP)
- ebXML
- FTP/SFTP
- Message queuing

- COBRA

The group determined that these methods are the most frequently used data exchange methods for IIS. After creating this original list, the group identified any major drawbacks associated with each method. For example, the group determined that using emails to transport messages had the major drawback that email requires encryption technology to meet security and privacy requirements. As a result of this initial, high-level review, the group concluded that only two methods, HTTP and SOAP, had no significant drawbacks and therefore warranted a more in-depth review. As such, this review guide will only focus on reviewing HTTP and SOAP.

Approach to Reviewing Data Exchange Methods

The Transport Layer Group assigned a specific individual to review each protocol and describe several key aspects of the protocol. These pieces of information included:

- General description of the protocol.
- The standards organization that supports development and maintenance of the protocol.
- Design goals of the protocol.
- Common uses of the protocol.
- Any underlying protocols needed to support the protocol.
- How the protocol addresses security.
- How the protocol meets compliance requirements.
- The entity or process that can certify the use of HTTP in IIS.
- How far along on the path to becoming a standard the technology is.
- The strengths of the protocol.
- The challenges associated with the protocol.

After this set of initial background information was collected, the assigned reviewer reviewed and rated (if applicable) the protocol in several areas. The rating scale ranged from 1 to 5, with 1 being the lowest and 5 being the highest.

- What operating systems/platforms does the protocol support?
- What programming languages does the protocol support?
- How easy is it for developers to design and program for this protocol?
- How easy is it for developers to maintain implementations that use this protocol?
- How does the protocol perform error handling?
- What is the technical support infrastructure like for the protocol?
- Provide an overall rating for the protocol?

The reviewer was then asked to provide the following:

- A sample of client code for this protocol
- Two case studies for the protocol
- Useful web site references for the protocol

The remainder of this document provides the details of those reviews, compares the two reviewed protocols with each other and discusses next steps for selecting and promoting use of one of these protocols as the data exchange layer for use in IIS.

Hypertext Transfer Protocol (HTTP) Review

General Information

General Description: In 2002, a subcommittee of the Committee for Immunization Registry Standards for Electronic Transactions (CIRSET) published a draft document entitled "Transport of Immunization HL7 Transactions over the Internet Using Secure HTTP." Joseph Rockmore of IBM Corporation, Andrey Yeatts of Scientific Technologies Corporation and Kevin Davidson of QS Technologies, Inc. authored the document. The standard proposed a set of conventions for transmitting HL7 transactions using familiar web protocols. Early implementers of IIS online transactions were attracted to the simplicity of the protocol. According to a survey conducted by the American Immunization Registry Association (AIRA), the HTTP/Post protocol is currently the most widely supported data exchange method.

Supporting Standards Organizations: The World Wide Web Consortium (W3C) and the Internet Engineering Task Force (IETF).

Protocol Design Goals: HTTP is designed to be a very generic protocol, making it adaptable to many purposes, including the transfer of immunization data in HL7. The HTTP/POST method is the method that requests that the destination server accept the entity enclosed in the request. The IIS implementation of HTTP/POST was designed for speed and simplicity.

Common Uses of the Protocol: The POST method is suitable for many purposes including providing a block of data, such as submitting the results of a form to a data handling process. The IIS implementation of HTTP/POST uses the same approach for transporting data that is used to transport data from a web page form, but uses specific data fields that are defined for authentication information and the HL7 payload. The IIS responds with a single character string comprised of the HL7 message, for example, an ACK response that acknowledges receipt of the message.

The conventions in this standard can apply to both the HTTP and HTTPS protocols, with HTTP being used only within private networks.

Underlying Protocols: HTTP uses the Transmission Control Protocol (TCP) to deliver traffic.

Security: HTTP/POST provides two methods for encryption. One method is an agency/user-id/password scheme; the second is authentication through a digital certificate. HTTP/POST manages authorization outside the protocol; presumably, authorization is tied to the presented agency and user information. HTTPS provides encryption when traffic passes over a public network, such as the Internet.

Compliance Requirements:
All web servers must meet the W3C specification. The draft Transport document cited in the General Description for HTTP provides a reference standard for how the POST message includes the immunization payload and authentication. It's important to note that currently, HL7 does not address the HTTP/POST protocol.

Certification Process:
Typically each IIS certifies systems using HTTP.

Status of HTTP as a Standard:
The HTTP/1.1 standard was officially released in January 1997 with improvements and updates to it released in June 1999.

Strengths
HTTP can be implemented quickly, crosses firewalls easily, is highly efficient, and requires minimal knowledge for a developer to use.

Challenges
When implemented with authentication via digital certificate, a PK1 infrastructure must exist. Also, valid HL7 error responses are difficult for server-level processes to generate.

Reviews and Rating

(Scale of 1 to 5, with 1 being poor and 5 being excellent)

Systems/Platform Protocol Supports (Rating of 5)
HTTP supports all industrial operating systems, including Windows, UNIX, Linux, Mac and Mainframe.

Programming Language Support (Rating of 5)
HTTP supports object-oriented languages such as Java, C++, C#, Python, and Delphi. Essentially HTTP supports any program that can submit a Web Form. Because HTTP essentially requires developer to work with character strings, support infrastructure is minimal.

Ease of Development (Rating of 5)
Because the standard was designed to be a very simplified protocol, it makes minimal requirements on developers.

Ease of Maintenance (Rating of 3)
Software maintenance of HTTP is relatively simple because of the inherent simplicity of the protocol. Problem determination can be difficult for the reasons discussed under Error handling, which follows.

Error Handling (Rating of 3)
Practical implementations of the protocol have proved difficult to debug, particularly because of the encrypted traffic. Both client and server implementations need to take care to provide detailed logging capability and robust error messaging. In addition, the HTTP/POST protocol requires that all error responses be valid HL7 messages. This is good because only HL7 needs to be spoken, but it creates difficulties when server-level errors have to be returned as valid HL7 (see Challenges section below).

Technical Support Infrastructure (Rating of 5)
Numerous online forums, technical documents, tutorials, commercial support contracts, and experts are available to support HTTP technology.

Overall Rating 4
The protocol in several installations proved easy to implement and operated robustly. There are no known significant problems with the protocol.

Sample Client Code

New York's Citywide Immunization Registry supports automated file transfers over HTTPS from the UNIX and Windows platforms and provides the software for automated transfers to its reporting and data exchange partners. The sending computer must meet the following software requirements:

- Have a supported Java virtual machine
- Have access to the CIR's Web site without proxy authentication from the sender
- Have access to the files that need to be transferred
- Be able to set up and run scheduled tasks (Windows) or cron jobs (UNIX)
- Under Windows, the program must be able to run a specific executable
- Under UNIX, a Python interpreter must be installed

Once those requirements are met, the automated file transfer software's configuration file needs to be populated with the location where the files are to be transferred from, and depending on the operating system platform, one following scripts is run on a daily basis.

Under Windows:
```
cd C:\cirAutomatedUploadFinal\wfrupload\javaUploadProgram
java -cp . TransferUPIFFileToCIR >>output.txt 2>>error.txt
```

Under UNIX:
```
cd /cirAutomatedUploadFinal/UNIX/javaUploadProgram
java -cp . TransferUPIFFileToCIR >>output.txt 2>>error.txt
```

Case Study: Marion County, Indiana

The Health and Hospital Corporation of Marion County serves the Indianapolis, Indiana metropolitan area, the 14th largest city in the US.

The health information system vendor for public health in Marion County was QS Technologies. The state IIS, CHIRP, was developed by Scientific Technologies Corporation.

The project began with a meeting of all the parties that would be involved in the project:

- The software vendor, QS Technologies

- The customer, Health and Hospital Corporation of Marion County
- The Indiana state immunization IIS, CHIRP
- The immunization registry software vendor, Scientific Technologies

These stakeholders reached a consensus on a two-way interface at this meeting. They agreed that that the clinic database would store only immunization histories from the IIS, and not demographic information. They also chose to reference the HL7 Immunization Registry Implementation Guide to implement the selected standard for data exchange—the HTTPS Immunization HL7 standard. The stakeholders agreed that the IIS would create staff records for providers from input HL7 messages rather than having to set them up in advance by phone or by fax.

The project plan was scheduled to begin testing after the August 2003 release of new IIS software from STC and to complete before the QS Technologies general release of its health information software, Insight 5.1, on December 1, 2003.

The stakeholders organized the primary point of contact between the technical staffs of the two vendors involved. STC provided a liaison with the Indiana IIS and QS technologies with the Health and Hospital Corporation of Marion County.

The stakeholders agreed that the data exchange protocol used would be the HTTPS protocol that resulted from work within the Committee on Immunization Standards for Electronic Transactions (CIRSET) with participation from both vendors. The stakeholders later decided to use the Digital Certificate option for the HTTPS protocol.

Overall, testing went well. Because both vendors were writing new software, the testing uncovered some minor bugs. However, two critical factors resulted in things going well:

1) The HTTPS protocol allowed immediate feedback on the success of message transmission. Because of the nature of the standard, it was not necessary for both vendors to schedule testing at the same time. Each vendor could test whenever they were ready. Sending a VXU and then being able to query it back for comparison allowed a round-trip test to be conducted from one end.
2) The trading partner agreement set expectations at the outset, so few surprises were encountered. The HL7 standard itself allowed painless arbitration over what was considered “correct.”

Informal test results over the Internet showed transactions averaging just over 2 seconds.

Case Study: New York City Department of Health and Mental Hygiene

New York Citywide Immunization Registry (CIR) has mandatory reporting of all immunization for children up to age 18, and for adults with informed consent.

CIR uses the Web File Repository (WFR), an HTTPS application developed by HLN Consulting, for all file-based reporting. This product was originally designed as a secure, web-based file transfer application, and was subsequently extended to meet immunization data file processing and to provide quality assurance feedback for each file submitted. The application automatically processes a NYC-specific Universal Provider Interface Format (UPIF) file for reporting, and partners who choose to submit HL7 files will have their files converted to the UPIF format by CIR quality assurance staff before the files are processed. Each file contains information on many patient immunization records.

WFR is also used for sending immunization information from CIR to data exchange partners using the NYC-specific data exchange interface (DEI) format. When a user submits a data exchange file that identifies the patients, a CIR staff member is notified, the staff member processes the file, and the results are uploaded into WFR for the requester.

File uploads to WFR can be performed manually over a web browser or automatically from a partner's computer. CIR provides software, including source code, that will automatically transfer the files to WFR from any software platform. This automated file transfer software is closely supported by CIR technical staff. Sometimes those using or planning to use the transfer software schedule site visits to install or support the software; however, many sites have installed and configured the software independently.

References

"Transport of Immunization HL7 Transactions over the Internet Using Secure HTTP"
http://www.immregistries.org/pdf/HL7_Secure_Transport_Ver1_0Sept02.pdf

Extensible Markup Language Protocol (XMLP) Review

General Information

General Description: Extensible Markup Language Protocol (XMLP), also known as Simple Object Access Protocol (SOAP), or Web Services, is a web-based technology that allows remote client systems to call server-based functionality over the World Wide Web. The technology enables disparate systems from anywhere in the world to seamlessly integrate over the Web with minimal technical challenges.

Supporting Standards Organizations: World Wide Web Consortium (W3C), HL7 and HITSB

Protocol Design Goals: XMLP/SOAP is the successor to remote procedure call (RPC) and Common Object Request Broker Architecture (CORBA) technologies. These technologies enabled client systems to call remote server functionality over dedicated networks. The evolution of these mature technologies along with the growth of the World Wide Web necessitated the addition of web-based technologies for doing the work that RPC and CORBA have performed for decades on private networks. The fundamental design contribution of XMLP/SOAP is the built-in ability to integrate disparate systems over large Wide Area Networks (WANs) like the Internet.

Common Uses of the Protocol: To integrate disparate systems over large WANs like the Internet. The Wikipedia entry for SOAP has the following example: a SOAP message could be sent to a web service enabled web site (for example, a house price database) with the parameters needed for a search. The site would then return an XML-formatted document with the resulting data (prices, location, features, etc). Because the data is returned in a standardized machine-parseable format, it could then be integrated directly into a third-party site.

Underlying Protocols: The typical XMLP/SOAP deployment uses Hypertext Transfer Protocol (HTTP) or its secure variant protocol (HTTPS) to send and receive requests over the Web. The Transmission Control Protocol (TCP) is typically used to deliver HTTP and HTTPS traffic.

Security: The software developer who implements an XMLP/SOAP service handles authentication, authorization, and accounting. Typical authentication methods include username, password, and X.509 certificate or an authentication key for server-to-server authentication. Accounting is usually made up of database and application server logs that show who called the service at specific days/times, with the specific parameters that were sent in the service call and the result of the call.

Because XMLP/SOAP is a web-based technology, no encryption code needs to be written to protect the contents of a message. By simply adding an SSL certificate to the server and requiring that all consumers of the message use the encrypted HTTPS protocol, the producer of the web service can ensure that all messages sent to and from the server are encrypted.

Compliance Requirements:
All XMLP/SOAP engines must meet the W3C specification. Currently, no national standards exist that define what an immunization registry XMLP/SOAP call should look like. However, if developed, such standards would be very helpful going forward.

Certification Process:
Because there are no immunization registry compliance requirements, there is no process to certify that a specific registry meets those requirements.

Status of HTTP as a Standard:
Version 1.2 of the SOAP standard became a W3C recommendation on June 24, 2003.

Strengths
The availability of good documentation and software developer support, along with ease of use are the greatest strengths of XMLP/SOAP.

Challenges
Programmer-defined objects cannot be passed between different languages, so programmers must use Strings or primitive types such as integers, characters, and arrays. For example, a programmer Visual Studio .net object cannot be passed to a Java client and vice-versa.

Reviews and Rating

(Scale of 1 to 5, with 1 being poor and 5 being excellent)

Systems/Platform Protocol Supports (Rating of 5)
XMLP/SOAP supports all industrial operating systems, including Windows, UNIX, Linux, Mac and Mainframe.

Programming Language Support (Rating of 4)
XMLP/SOAP supports object-oriented languages such as Java, C++, C#, Python, and Delphi. Client support for Microsoft's .net, Sun Microsystems Java, and IBM Java are built in to the runtime environment. The only weaknesses for XMLP/SOAP programming language support is for languages that are not object-oriented and older languages that were written before XMLP/SOAP was designed.

Ease of Development (Rating of 4)
Server-side development for XMLP/SOAP is intuitive and natural for an object-oriented programmer. Client-side development for XMLP/SOAP requires pulling of Web Services Description Language (WSDL) to communicate with the server's interface.

Ease of Maintenance (Rating of 4)
Well-documented open-source and proprietary application servers are available. Application server logs assist the individual maintaining XMLP/SOAP implementations in narrowing issues down to their cause. Deployment of redundant systems can be as simple as adding an additional server.

Error handling (Rating of 3)
Some web proxy products catch specifically targeted web service exceptions and send less specific exceptions to the requestor of the service. This issue can be worked around by combining return values and modifying reference parameters.

Technical Support Infrastructure (Rating of 5)
Numerous online forums, technical documents, tutorials, commercial support contracts, and experts are available to support the XMLP/SOAP technology.

Overall Rating 4
At the time this review was completed, XMLP/SOAP is the best technology available for real-time bi-directional communication of immunization information between healthcare provider EHRs and immunization information systems.

Sample Client Code

To submit a vaccine record update (VXU) HL7 message to the New York Citywide Immunization Registry, a partner system would write like the code sample below:

```
public static void main(String[] args)
{
        String serviceEndPoint = "https://hostname/VaccineService";
        String userName = "client user name";
        String password = "client password";

        String responseMessage = null;
        try
        {
            //Initialize the service
            VaccineServiceStub service = new
            VaccineServiceStub(serviceEndPoint);

            //Set the HTTP authentication properties
            HttpTransportProperties.Authenticator auth = new
            HttpTransportProperties.Authenticator();
            auth.setUsername(userName);
            auth.setPassword(password);

            //Get the Service Options and set the Authentication //Properties
            on the Options
            Options opts = service._getServiceClient().getOptions();
            opts.setManageSession(true);
            opts.setProperty(HTTPConstants.AUTHENTICATE, auth);
            opts.setProperty(HTTPConstants.CHUNKED, Boolean.FALSE);

            // Create a new instance of the VXU Request object
            SubmitPatientImmRequestNoKey request =
            SubmitPatientImmRequestNoKey.class.newInstance();
            SubmitPatientImmResponse response = null;

            //Create the Message and set the VXU Message on the message
            //object
            String vxuInMessage =
"MSH|^~\\&|PATIENTS1ST1.1|1234567|||20080424162946||VXU^V04|578438|T|2.3.1|||
|AL|\r" +
"PID|||531151424^^^^LR~BB77777B^^^^MA~221345671^^^^MR||CARRY^JOHN^J^^^^
|WALTERS^^^^^^M|19991125|M| CARRIE^JOHNNY^^^^^A|2106-3^WHITE^HL70005|1907
CRUMPTON ROAD ^APT 3B^JAMAICA^NY^11423^^
||^^^^^617^5551212||EN^ENGLISH^HL70296||||||N^NOT HISPANIC OR
LATINO|11116|N|\r" +
"NK1||JONES^MARY^ANN|MTH^MOTHER^HL70063||^^^^^212^5218118|^^^^^212^7771212^49
7||||||||||19781115|\r" + "PV1|||||||||||||||||||V02\r" +
"RXA|||20000607||08^HEP B, ADOLESCENT OR PEDIATRIC^CVX||||03^HISTORICAL
INFORMATION - FROM PARENT'S WRITTEN
RECORD^NIP0001|6145123^JONES^LISA^A^^^^^^^^^OEI
|^^^1234567||||W2348796456|20020731|MSD^MERCK^MVX||||A\r" +
"RXA|||20040607||100^PNEUMOCOCCAL
CONJUGATE^CVX||||04|9412390^SMITH^BOB^L^^^^^^^^^OEI
|^^^1234567||||W2348796456|20080820|MSD^MERCK^MVX| |||A";

            Message message = new Message();
            message.setMessage(vxuInMessage);

            //Set the Message on the Request
            request.setVxuInMessage(message);

            //Send it and get the response
            response = service.SubmitPatientImmRecordNoKey(request);
            responseMessage = response.getVxuOutMessage().getMessage();
        }
        catch (Exception e)
        {
            responseMessage = trapException(e);
        }
        finally
        {
            System.out.println("response=" + responseMessage);
        }
}
```

To submit a vaccine record query (VXQ) HL7 message to the New York Citywide Immunization Registry, a partner system would write like the code sample below:

```
public static void main(String[] args)
{
        String serviceEndPoint = "https://hostname/VaccineService";
        String userName = "client user name";
        String password = "client password";
        String responseMessage = null;

        try
        {
            // Initialize the service
            VaccineServiceStub service = new
            VaccineServiceStub(serviceEndPoint);

            // Set the HTTP authentication properties
            HttpTransportProperties.Authenticator auth = new
            HttpTransportProperties.Authenticator();
            auth.setUsername(userName);
            auth.setPassword(password);

            // Get the Service Options and set the Authentication
            //Properties on the Options
            Options opts = service._getServiceClient().getOptions();
            opts.setManageSession(true);
            opts.setProperty(HTTPConstants.AUTHENTICATE, auth);
            opts.setProperty(HTTPConstants.CHUNKED, Boolean.FALSE);

            // Create a new instance of the VXQ Query Request object
            QueryPatientImmRequestNoKey request =
            QueryPatientImmRequestNoKey.class.newInstance();
            QueryPatientImmResponse response = null;

            // Create the Message and set the VXQ Message on the message
object
            String vxqInMessage =
"MSH|^~\\&|PATIENTS1ST1|1234567||STFAC00|20100301134600||VXQ^V01|20100301GA40
|T|2.3.1|||NE|ER|\r" +
"QRD|20100301134602|R|I|20100301GA05|||627210336^HAMMERHEAD^FRED^^^^^^^^^^LR
|\r" + "QRF|||||~19900101\r" + "ZGR|M";

            Message message = new Message();
            message.setMessage(vxqInMessage);

            // Set the Message on the Request
            request.setVxqInMessage(message);

            // Send the request and get the response
            response = service.QueryPatientImmRecordNoKey(request);
            responseMessage = response.getVxqOutMessage().getMessage();
        }
        catch (Exception e)
        {
            responseMessage = trapException(e);
        }
        finally
        {
            System.out.println("response=" + responseMessage);
        }

}
```

Case Studies

One of the goals of Web Services in Immunization Information Systems is the bi-directional exchange of immunization information between the Health Department and the healthcare provider in real-time.

New York's Citywide Immunization Registry released a production implementation of its real-time Web Services to accept HL7 VXQ and VXU messages in March, 2009. Before production rollout, the test environment was made available to all interested partners who wanted to begin development. A number of EMR/EHR vendors expressed interest in these services, but none implemented the integration in production, many did not enter a testing phase, and most did not begin development.

There are a number of teaching hospitals in New York City that also have graduate programs in Public Health Informatics. In July, 2009 Columbia Presbyterian's EZVAC registry implemented a production integration with the CIR VXQ service. Columbia has historically reported all of its vaccinated patients on text files using https post, but was not previously able to query this many patients in real time.

Columbia is using the CIR VXQ service two ways:

1. Quarterly queries of all patients in the EZVAC system to provide Columbia with the most accurate immunization coverage rate
2. Real-time queries of single CIR patients to give healthcare providers the latest immunization information from the public health database

At the end of February, 2010, Columbia had made 365,803 VXQ requests using the Web Service. Their next step is to start development using the VXU service for real-time immunization reporting and they are also considering using the decision support recommendations returned by the VXQ service.

References

Apache Axis open-source implementation of XMLP/SOAP for Java and C++:
http://ws.apache.org/axis/

Technical specification documents at W3C:
http://www.w3.org/2000/xp/Group/

Next Steps

The recent emphasis on Meaningful Use in health information technology has spurred initiatives around data exchange and system interoperability standards. As these initiatives result in recommended or required protocols for application layer standards, additional protocol descriptions will be added to this document.