EHR-IIS Interoperability Enhancement Project

Transport Layer Protocol Recommendation
Formal Specification

Version 1.2
September 4, 2015

Transport Layer Expert Panel
EHR-IIS Interoperability Enhancement Project
Immunization Information Systems Support Branch (IISSB)
National Center for Immunization and Respiratory Disease (NCIRD)
Centers for Disease Control and Prevention (CDC)

Table of Contents

Tables and Figures

1 Background

The Transport Layer Expert Panel has recommended a SOAP-based transport methodology for health system-to-health system HL7 immunization messaging interoperability. This document describes the underlying transport, security, and SOAP operations of the recommended approach.

The scope of this document is limited to transport, security, and SOAP operations, parameters, and faults for SOAP-based HL7 transmissions to an IIS. Although the transport layer is message agnostic, the expected use of the methodology is to send HL7 version 2.x messages (e.g.: 2.3.1, 2.5.1, etc…) presently used in the Immunization Information System (IIS) setting.

The web service specification described in this document is designed to transmit single HL7 messages synchronously, i.e., a single HL7 message is transmitted and a response is generated immediately. Batch messages and asynchronous responses are out of scope for this specification, but are discussed in more detail in the appendix (see Section 6).

2 Transport

The Sender and Receiver SHALL conform to SOAP 1.2 over HTTPS (HTTP over TLS 1.1 or newer) using the authentication and web service specification described in the following subsections.

3 Security

Transport layer encryption is provided by TLS; authentication and authorization of the sender must be performed by the receiver either using username and password credentials passed as part of the SOAP operations (see Section 4.4), or using client certificate authentication via TLS, or both. The authentication and authorization methods supported by a receiving IIS are typically published in a local HL7 implementation guide for the IIS.

4 SOAP Web Service

4.1 Actors

There are two actors in the sending of HL7 messages via SOAP in the IIS setting:

1. **Sender** – typically an Electronic Health Record system (EHR-S) operated by an immunization provider, or an entity acting on behalf of an immunization provider. The sender operates a SOAP client to send HL7 messages to an IIS.
2. **Receiver** – typically an IIS operated by a state or local health department. The receiver operates a SOAP Web Service to receive HL7 messages from the sender.

4.2 Workflow

The general workflow for sending an HL7 message via SOAP to an IIS follows:

1. Sender tests connectivity to IIS
2. Sender composes and sends HL7 message
3. Receiver accepts HL7 message and sends HL7 response
4. Sender accepts HL7 response and any faults

4.3 Operations

The following operations are provided by the IIS SOAP Web Service to support the workflow:

Table 1 Operations

Operation	Purpose
connectivityTest	To test connectivity; to verify that the SOAP Web Service is accessible.
submitSingleMessage	To submit an HL7 version 2.x message (e.g.: 2.3.1, 2.5.1) to an IIS.

4.4 Parameters

Each operation has one or more input and output parameters:

Operation: **connectivityTest**

Table 2 Connectivity Test Operation Parameters

Parameter	Input/Output	Data type	Description
echoBack	Input	String	Data to be sent back by the connectivity test.
return	Output	String	Data sent back by the test. The returned string should include the original text sent in by the sender. Other text may be prepended or appended.

Operation: **submitSingleMessage**

Table 3 Submit Single Message Operation Parameters

Parameter	Input/Output	Data type	Description
username	Input	String	IIS username
password	Input	String	IIS password
facilityID	Input	String	IIS Facility ID
hl7Message	Input	String	HL7 version 2.x message (e.g.: 2.3.1, 2.5.1, etc…) intended for IIS

Parameter	Input/Output	Data type	Description
Return	Output	String	HL7 version 2.x response (e.g.: 2.3.1, 2.5.1, etc…) from IIS

NOTE: The username, password, and facilityID parameters are technically optional, but are heavily used by IIS across the nation for authentication. These parameters, if used, are defined by the IIS and provided to the sender prior to initiating HL7 transmissions. Given their heavy usage the following system capabilities have been defined.

- A sender SHALL have the ability to provide any combination of username, password, and/or facilityID where required by receiver.
- A receiver MAY require the use any combination of username, password, and/or faciltyID.

The hl7Message and return parameter must contain the appropriate HL7 message as defined by the Implementation Guide for Immunization Data Transactions using Version 2.x (e.g.: 2.3.1, 2.5.1, etc…) of the Health Level Seven (HL7) Standard Protocol, and any local IIS HL7 implementation guides.

4.5 Faults

The SOAP Fault element is used to indicate error messages related to the SOAP operations and to carry detailed information within a SOAP message regarding the error.

There are four types of SOAP Faults in the IIS SOAP Web Service:

1. **UnsupportedOperationFault_Message** – generated if the sender attempts to request an operation that is not part of the IIS SOAP Web Service (See Section 4.3).
    a. A receiver MAY have the ability to throw this fault.
    b. A sender SHALL have the ability to catch this fault.
2. **SecurityFault_Message** – generated if the authentication credentials supplied in the **submitSingleMessage** operation are not validated.
    a. A receiver SHALL have the ability to throw this fault.
    b. A sender SHALL have the ability to catch this fault.
3. **MessageTooLargeFault_Message** – generated if the hl7Message parameter of the **submitSingleMessage** operation is too large. The maximum length (e.g.: number of messages, number of characters, etc…) should be specified by the IIS and provided to the sender prior to initiating HL7 transmissions.
    a. A receiver MAY have the ability to throw this fault.
    b. A sender SHALL have the ability to catch this fault.
4. **UnknownFault_Message** – Any SOAP fault that does not fit into one of the above three SOAP Fault categories will be returned as an "unknown" fault.
    a. A receiver MAY have the ability to throw this fault.
    b. A sender SHALL have the ability to catch this fault.

Each type of SOAP Fault contains the following parameters:

Table 4 SOAP Fault Parameters

Parameter	Input/Output	Data type	Description
Code	Output	Integer	SOAP Fault code number, intended for automated use by client software to identify the fault.
Reason	Output	String	SOAP Fault reason, intended to be a human-readable explanation of the error that caused the fault.
Detail	Output	String	Detailed explanation of fault.

Fault code numbers should be specified by the IIS and provided to the sender prior to initiating HL7 transmissions.

4.6 Formal Specification

The formal specification for the IIS SOAP Web Service is contained in the following Web Services Definition Language (WSDL) document.

4.6.1 Header

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
             xmlns:wsp="http://www.w3.org/ns/ws-policy"
             xmlns:wsp1_2="http://schemas.xmlsoap.org/ws/2004/09/policy"
             xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"
             xmlns:wsaw="http://www.w3.org/2005/08/addressing"
             xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
             xmlns:tns="urn:cdc:iisb:2011"
             xmlns:xsd="http://www.w3.org/2001/XMLSchema"
             xmlns="http://schemas.xmlsoap.org/wsdl/"
             targetNamespace="urn:cdc:iisb:2011"
             name="IISServiceNew">
```

4.6.2 Schema for types

```
<!-- schema for types -->
<types>
  <xsd:schema elementFormDefault="qualified" targetNamespace="urn:cdc:iisb:2011">

    <xsd:complexType name="connectivityTestRequestType">
      <xsd:sequence>
        <xsd:element name="echoBack" type="xsd:string" minOccurs="1" maxOccurs="1" nillable="true"/>
      </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="connectivityTestResponseType">
      <xsd:sequence>
        <xsd:element name="return" type="xsd:string" minOccurs="1" maxOccurs="1" nillable="true"/>
      </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="submitSingleMessageRequestType">
      <xsd:sequence>
        <xsd:element name="username" type="xsd:string" minOccurs="0" maxOccurs="1" nillable="true"/>
        <xsd:element name="password" type="xsd:string" minOccurs="0" maxOccurs="1" nillable="true"/>
        <xsd:element name="facilityID" type="xsd:string"  minOccurs="0" maxOccurs="1" nillable="true"/>
        <xsd:element name="hl7Message" type="xsd:string" minOccurs="1" maxOccurs="1" nillable="true"/>
      </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="submitSingleMessageResponseType">
      <xsd:sequence>
        <xsd:element name="return" type="xsd:string" minOccurs="1" maxOccurs="1" nillable="true"/>
      </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="soapFaultType">
      <xsd:sequence>
        <xsd:element name="Code" type="xsd:integer" minOccurs="1"/>
        <xsd:element name="Reason" type="xsd:string" minOccurs="1"/>
        <xsd:element name="Detail" type="xsd:string" minOccurs="1"/>
      </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="UnsupportedOperationFaultType">
      <xsd:sequence>
        <xsd:element name="Code" type="xsd:integer" minOccurs="1"/>
        <xsd:element name="Reason" fixed="UnsupportedOperation"/>
        <xsd:element name="Detail" type="xsd:string" minOccurs="1"/>
      </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="SecurityFaultType">
      <xsd:sequence>
        <xsd:element name="Code" type="xsd:integer" minOccurs="1"/>
        <xsd:element name="Reason" fixed="Security"/>
        <xsd:element name="Detail" type="xsd:string" minOccurs="1"/>
      </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="MessageTooLargeFaultType">
```

```
            <xsd:sequence>
              <xsd:element name="Code" type="xsd:integer" minOccurs="1"/>
              <xsd:element name="Reason" fixed="MessageTooLarge"/>
              <xsd:element name="Detail" type="xsd:string" minOccurs="1"/>
            </xsd:sequence>
          </xsd:complexType>

          <xsd:element name="connectivityTest" type="tns:connectivityTestRequestType"/>
          <xsd:element name="connectivityTestResponse" type="tns:connectivityTestResponseType"/>
          <xsd:element name="submitSingleMessage" type="tns:submitSingleMessageRequestType"/>
          <xsd:element name="submitSingleMessageResponse" type="tns:submitSingleMessageResponseType"/>
          <xsd:element name="fault" type="tns:soapFaultType"/>
          <xsd:element name="UnsupportedOperationFault" type="tns:UnsupportedOperationFaultType"/>
          <xsd:element name="SecurityFault" type="tns:SecurityFaultType"/>
          <xsd:element name="MessageTooLargeFault" type="tns:MessageTooLargeFaultType"/>

        </xsd:schema>
      </types>
```

4.6.3 Message definitions

```
      <!-- Message definitions -->
      <message name="connectivityTest_Message">
        <documentation>connectivity test request</documentation>
        <part name="parameters" element="tns:connectivityTest" />
      </message>

      <message name="connectivityTestResponse_Message">
        <documentation>connectivity test  response</documentation>
        <part name="parameters" element="tns:connectivityTestResponse" />
      </message>

      <message name="submitSingleMessage_Message">
        <documentation>submit single message request.</documentation>
        <part name="parameters" element="tns:submitSingleMessage" />
      </message>

      <message name="submitSingleMessageResponse_Message">
        <documentation>submit single message response</documentation>
        <part name="parameters" element="tns:submitSingleMessageResponse" />
      </message>

      <message name="UnknownFault_Message">
        <part name="fault" element="tns:fault"/>
      </message>

      <message name="UnsupportedOperationFault_Message">
        <part name="fault" element="tns:UnsupportedOperationFault"/>
      </message>

      <message name="SecurityFault_Message">
        <part name="fault" element="tns:SecurityFault"/>
      </message>

      <message name="MessageTooLargeFault_Message">
        <part name="fault" element="tns:MessageTooLargeFault"/>
      </message>
```

4.6.4 Operation/transaction declarations

```
      <!-- Operation/transaction declarations -->
      <portType name="IIS_PortType">
        <operation name="connectivityTest">
          <documentation>the connectivity test</documentation>
          <input message="tns:connectivityTest_Message" wsaw:Action="urn:cdc:iisb:2011:connectivityTest"/>
          <output message="tns:connectivityTestResponse_Message"
wsaw:Action="urn:cdc:iisb:2011:connectivityTestResponse"/>
          <fault name="UnknownFault" message="tns:UnknownFault_Message"/> <!-- a general soap fault -->
          <fault name="UnsupportedOperationFault" message="tns:UnsupportedOperationFault_Message"/> <!-- The
UnsupportedOperation soap fault -->
        </operation>

        <operation name="submitSingleMessage">
          <documentation>submit single message</documentation>
          <input message="tns:submitSingleMessage_Message" wsaw:Action="urn:cdc:iisb:2011:submitSingleMessage"/>
          <output message="tns:submitSingleMessageResponse_Message"
wsaw:Action="urn:cdc:iisb:2011:submitSingleMessageResponse"/>
          <fault name="UnknownFault" message="tns:UnknownFault_Message"/> <!-- a general soap fault -->
          <fault name="SecurityFault" message="tns:SecurityFault_Message"/>
          <fault name="MessageTooLargeFault" message="tns:MessageTooLargeFault_Message"/>
```

```
        </operation>
      </portType>
```

4.6.5 SOAP 1.2 Binding

```
      <!-- SOAP 1.2 Binding -->
      <binding name="client_Binding_Soap12" type="tns:IIS_PortType">
        <soap12:binding style="document" transport="http://schemas.xmlsoap.org/soap/http" />
        <operation name="connectivityTest">
          <soap12:operation soapAction="urn:cdc:iisb:2011:connectivityTest" />
          <input><soap12:body use="literal" /></input>
          <output><soap12:body use="literal" /></output>
          <fault name="UnknownFault"><soap12:fault use="literal" name="UnknownFault"/></fault>
          <fault name="UnsupportedOperationFault"><soap12:fault use="literal"
name="UnsupportedOperationFault"/></fault>
        </operation>
        <operation name="submitSingleMessage">
          <soap12:operation soapAction="urn:cdc:iisb:2011:submitSingleMessage" />
          <input><soap12:body use="literal" /></input>
          <output><soap12:body use="literal" /></output>
          <fault name="UnknownFault"><soap12:fault use="literal" name="UnknownFault"/></fault>
          <fault name="SecurityFault"><soap12:fault use="literal" name="SecurityFault"/></fault>
          <fault name="MessageTooLargeFault"><soap12:fault use="literal" name="MessageTooLargeFault"/></fault>
        </operation>
      </binding>
```

4.6.6 Service definition and footer

```
      <!-- Service definition -->
      <service name="client_Service">
        <port binding="tns:client_Binding_Soap12" name="client_Port_Soap12">
          <soap12:address location="http://localhost/WebApp/IISService" />
        </port>
      </service>
    </definitions>
```

5 Document Management

Table 5 Document Management

Date	Changed By	Comments	Version #
8/25/2011	Transport Layer Expert Panel	Initial Version	1.0
6/4/2014	E. Larson	Added Appendix B to document the end-of-line terminator disagreement between standards.	1.1
9/3/2015	E. Larson	Added conformance statements to align with WSDL. Added clarifying statements where appropriate based on community input. No changes were made to the WSDL (Section 4.6)	1.2

6 Appendix A: SOAP-Based Asynchronous/Batch Exchange

6.1 Overview

When recommending a transport layer for health information system to IIS interoperability, it was the goal of the transport layer expert panel to address different processing scenarios and payload sizes, including synchronous and asynchronous (and/or batch) exchanges. Through a detailed, consensus-based research process, the panel came to the conclusion that SOAP web services was the best choice to handle all of the current and future needs of IIS.

In order to truly promote interoperability, the panel recognized it was important to also define a standard interface for the recommended SOAP transport layer. The immediate need was for synchronous HL7 message exchange, so that interface was defined first.

Through investigation and detailed meetings, the unique requirements for asynchronous and/or batch processing were also discovered. The remainder of this appendix will discuss asynchronous and/or batch processing through SOAP, the specific differences between defining a standard for synchronous and asynchronous exchanges, and the panel's action plan.

6.2 Asynchronous Exchange and SOAP

From a purely technical standpoint, SOAP has no limitations preventing it from supporting asynchronous processing. Further, through the use of Message Transmission Optimization Mechanism (MTOM), the size of the payload being sent across the wire is not an issue. Today, several IIS, including Nevada, Massachusetts, Arizona, and Kansas, provide the ability to submit large payloads for processing through a SOAP web service.

While several IIS have asynchronous and/or batch processing via a SOAP web service, most of them have unique solutions which integrate their IIS batch processing and business processes into their SOAP web service definition. This creates a challenge when trying to define a standard interface usable by all trading partners.

However, it was acknowledged from the outset that the recommended transport layer was not intended to replace existing functional interfaces. With this in mind, and the large majority of asynchronous and/or batch exchanges already functional, the need for a SOAP-based standard interface for asynchronous exchange is likely small. It is acknowledged that the need exists, but it is assumed to be small in comparison to the need for a synchronous standard interface.

6.3 Defining a Standard Interface

Defining a standard interface to submit a batch payload for asynchronous processing is largely trivial through SOAP. In fact, the expert panel had basic consensus on a submission operation through the use of MTOM. The difficulty in a standard interface for asynchronous process exists after the batch payload has been submitted for

processing. Once a batch payload is in the hands of an IIS, it can take on multiple status codes defining the condition or state of the payload. These status codes are unique to each IIS.

When the sending system wants to check on the submission, each IIS may have a unique response. Without an already defined standard set of status codes, or an agreement across all IIS on what these codes should be, it becomes a futile effort to assume the correct solution. A simple set of status codes might be: "working," "finished," "not found," and "error." However, this may not be sufficient for all IIS.

Further, it is unknown at this time what each IIS uses to uniquely identify a submission and how that might be consistently messaged through a standard interface. That is, if the sending system cannot receive and process the unique identifier for a submission, it can never ask for an update or the response payload. While this problem isn't as challenging to solve as the status code problem, it is a known condition at this time.

6.4 Action Plan

As noted above, the panel acknowledges asynchronous and/or batch processing still has its place in interoperability and is easily accomplished from a technical standpoint using SOAP. However, there is no need to replace processes that are already working well. As a result, the panel is focusing its work on the immediate need to define a national standard interface for synchronous transmissions of HL7 messages. If there is a demonstrated need for a national standard interface for asynchronous processing as well, the panel will engage the interested parties and address the need through a consensus-based approach.

7 Appendix B: Implementation Notes

7.1 SOAP, HL7, and End-of-Line Terminators

A subtle, but important, disagreement between the HL7 V2 standard and an underlying SOAP standard was uncovered during testing with a new provider in Rhode Island in spring 2014.

The issue has to do with end-of-line terminators.

- The HL7 standard dictates that all lines shall end with a carriage return (i.e.: ASCII 13, \r, or #xD).
- The underlying XML standard used by SOAP dictates that all end-of-line terminators should be normalized to a line feed (i.e.: ASCII 10, \n, or #xA).

As such, it is possible that the carriage returns in an HL7 message could be (as proven in Rhode Island) converted to line feeds through SOAP transmission. Depending upon your HL7 parser, this could be problematic.

As of March 2014, 26 IIS were either in testing or production with the CDC WSDL so it was important to consider the ramifications of any suggested resolutions. At this time the suggestion is a resolution on the IIS side. This will eliminate the need to roll-out an updated version of the WSDL and most importantly will not require changes by providers.

The suggested resolution is one of two approaches.

1. Prior to calling the HL7 parser in your IIS, perform a quick find/replace to ensure carriage returns are present
2. Adjust the HL7 parser to allow more than just carriage returns to mark the end-of-line terminator