Nationwide Health Information Network (NHIN)

Messaging Platform
Specification

V 2.0

1/29/2010

Contributors

Name	NHIO Represented	Organization
Neel Phadke	CareSpark	
Vinod Muralidhar		
Taha Abdul-Basser	MedVirginia	
Ashish Shah	Delaware	
Craig Miller	NHIN-C	Vangent
Erik Rolf	CareSpark	Deliberare
Thomas Silvious	New York	GSI Health
Eric Heflin	DHIN	Medicity
David L. Riley	Fed NHIO	FHA
Rich Kernan	ONC/NHIN	Deloitte
Jackie Key	ONC/NHIN	Deloitte

Document Change History

Version	Date	Changed By	Items Changed Since Previous Version
0.1		Craig Miller	Initial Draft
1.0		M&C Subgroup	Updates
1.1		M&C Subgroup	Updates
1.2		M&C Subgroup	Updates
1.3		M&C Subgroup	Updates
1.4		M&C Subgroup	Sub group feedback incorporated
1.5		M&C Subgroup	Updates
1.6		Craig Miller	Updates
1.7		Neel Phadke	Formatting
1.8		Craig Miller	Updates based on T&S Workgroup feedback
1.8.1		Deborah Lafky	Formatting, preparation for HITSP review
1.9		Craig Miller	Provided explanatory text re SOAP versioning, included specification for SOAP messaging style and SOAP faults, included specification for WS-Base Notification
1.9.1	01/30/2009	David L. Riley	Re-formatted and edited to prepare for publication
1.9.2	04/02/2009	Craig Miller	Updated to include change to SOAP 1.2 and WS-I Basic Profile 2.0
1.9.3	04/13/2009	Craig Miller	Minor text edits, updated
1.9.4	06/22/2009	Craig Miller	Added Reliable Messaging
1.9.5	06/25/2009	Craig Miller	Updated Reliable Messaging notes, corrected version of UDDI
1.9.6	06/25/2009	Craig Miller	Minor text edits for consistency
1.9.7	09/01/2009	Craig Miller	Initial updates for policy assertions
1.9.8	09/16/09	Eric Heflin, Erik Rolf	PKI certificate revocation checking additions
2.0	1/29/2010	Craig Miller, Rich Kernan, Jackie Key	Updates based on work group feedback. Applied consistent formatting/language and enhanced clarity.

Document Approval

Version	Date	Approved By	Role
2.0	1/25/2010	NHIN Technical Committee	

Table of Contents

1 Preface

1.1 Introduction

The Nationwide Health Information Network (NHIN) Foundation specifications define the primary set of services and protocols needed to establish a messaging, security, and privacy foundation for the NHIN. It is upon this foundation that the functional set of NHIN web service interfaces operates.

This specification does not describe a web service interface. Instead, it defines the base set of messaging standards and web service protocols which must be implemented by each Health Information Organization (HIO) participating as nodes on the NHIN. The purpose of this specification is to define a common messaging platform to be used by all NHIN nodes in order to promote secure information exchange. This Messaging Platform specification is foundational to the NHIN and applies to every message to be exchanged among NHIN nodes.

1.2 Intended Audience

The primary audiences for NHIN Specifications are the individuals responsible for implementing software solutions that realize these interfaces at Health Information Organizations (HIOs) who are, or seek to be, nodes on the NHIN network. HIOs which act as nodes on the NHIN are termed NHIOs. This specification document is intended to provide an understanding of the context in which the web service interface is meant to be used, the behavior of the interface, the Web Services Description Language (WSDLs) used to define the service, and any Extensible Markup Language (XML) schemas used to define the content.

1.3 Business Needs Supported by this Specification

The NHIN requires a common, standards based platform for secure and reliable exchange of messages between NHIOs in a manner that is independent of specific operating system or programming language frameworks, as well as a technical trust fabric to support the DURSA.

This specification is designed to address the need for a common set of security and messaging protocols to enable the interoperable, secure exchange of health information across the NHIN. Further, the Messaging Platform is required to support two of the NHIN's central architectural principles:

Use the Public Internet – The NHIN is not a separate physical network, but a set of protocols and standards that enable secure health information exchange via the public Internet.

Platform neutral – The NHIN has adopted a stack (web services) that can be implemented using many operating systems and programming languages.

Together with the NHIN Authorization Framework, this specification is part of the NHIN's messaging, security, and privacy foundation. All NHIN service interface specifications assume this foundation.

1.4 Referenced Documents and Standards

The following documents and standards were referenced during the development of this specification. Deviations from or constraints upon these standards are identified below.

1) **Org/SDO name:** WS-I

   Reference # / Spec Name: Basic Profile

   Version #: v2.0

   Underlying Specs:

- Simple Object Access Protocol (SOAP) v1.2
- SOAP Message Encoding Style
- SOAP Faults
- Hypertext Transfer Protocol (HTTP) v1.1
- WS-Addressing v1.0
- WS-BaseNotification v1.3
- Message Transmission Optimization Mechanism (MTOM) binding for SOAP v1.0
- Web Services Description Language (WSDL) v1.1
- XML Schema v1.0
- Universal Discovery and Description Interface (UDDI) v3.0.2

NHIN Deviations or Constraints: Any specific NHIN deviations or constraints upon this or any underlying specifications are described in section 2.2.1 “Basic Profile”.

Link: http://www.ws-i.org/Profiles/BasicProfile-2_0(WGD).html

2) **Org/SDO name:** WS-I

Reference # / Spec Name: Security Profile

Version #: v1.1

Underlying Specs:

- Transport Layer Security v1.0
- XML Signature v1.0
- Web Services Description Language (WSDL) v1.1
- Symmetric Encryption Algorithm and Key Length AES 128-bit
- X.509 Token Profile v1.0
- Attachment Security v1.0

NHIN Deviations or Constraints: Any specific NHIN deviations or constraints upon this or any underlying specifications are described in section 2.2.2 “Security Profile”.

Link: http://www.ws-i.org/Profiles/BasicSecurityProfile-1.1.html

3) **Org/SDO name:** Internet Engineering Task Force (IETF)

Reference # / Spec Name: Public Key Infrastructure for X.509 Certificates (PKIX) RFC2560; Online Certificate Status Protocol (OCSP)

Version #: June 1999

Underlying Specs:

NHIN Deviations or Constraints: Any specific NHIN deviations or constraints upon this specification are described in section 3.3.2 “Certificate Verification and Revocation”.

Link: http://www.ietf.org/rfc/rfc2560

4) **Org/SDO name:** Internet Engineering Task Force (IETF)

Reference # / Spec Name: PKIX RFC 5280; Certificate Revocation List (CRL) Profile

Version #: May 2008

Underlying Specs:

NHIN Deviations or Constraints: Any specific NHIN deviations or constraints upon this specification are described in section 3.3.2 “Certificate Verification and Revocation”.

Link: http://tools.ietf.org/html/rfc5280

Link: http://www.ietf.org/rfc/rfc2560

5) **Org/SDO name:** OASIS

Reference # / Spec Name: WS-Reliable Messaging

Version #: v 1.2

NHIN Deviations or Constraints:

Underlying Specs:

Link: http://docs.oasis-open.org/ws-rx/wsrm/200702/wsrm-1.2-spec-cd-01.pdf

6) **Org/SDO name:** W3C

Reference # / Spec Name: WS- Policy

Version #: v 1.5

NHIN Deviations or Constraints:

Underlying Specs:

Link: http://www.w3.org/TR/2007/REC-ws-policy-20070904/

7) **Org/SDO name:** W3C

Reference # / Spec Name: WS- Policy Attachments

Version #: v 1.2

NHIN Deviations or Constraints:

Underlying Specs:

Link: http://www.w3.org/Submission/2006/SUBM-WS-PolicyAttachment-20060425/

8) **Org/SDO name:** W3C

Reference # / Spec Name: WS- Policy Framework

Version #: v 1.2

NHIN Deviations or Constraints:

Underlying Specs:

Link: http://www.w3.org/Submission/2006/SUBM-WS-Policy-20060425/

9) **Org/SDO name:** OASIS

Reference # / Spec Name: WS- Security Policy

Version #: v 1.2

NHIN Deviations or Constraints:

Underlying Specs:

Link: http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200512/ws-securitypolicy-1.2-spec-cd-01.pdf

1.5 Relationship to Other NHIN Specifications

This specification is related to other NHIN specifications as described below.

- **Authorization Framework** – defines the exchange of metadata used to characterize each NHIN request. The purpose of that exchange is to provide the responder with the information needed to make an authorization decision for the requested function. Each initiating message must convey information regarding end user attributes and authentication using SAML 2.0 assertions.

  Together with the Authorization Framework, the Messaging Platform defines the foundational messaging, security and privacy mechanisms for the NHIN.

The Messaging Platform is not a service, nor is it specifically related as part of a transaction to any of the NHIN services. Rather, it describes the common messaging and security protocols which apply to all NHIN messages and information exchange patterns.

2 Transport Description

2.1 Definition

The Messaging Platform describes the common web service protocols that must underlie every message transmitted between NHIOs. These specifications represent a common messaging and security platform for all other NHIN services.

The Messaging Platform describes the transport rather than the interface specifications as Messaging Platform consists of the underlying common elements of message transport rather than individual programming interfaces that can be invoked as web services. Along with the Authorization framework, this specification forms the NHIN's messaging, security, and privacy foundation.

2.2 Design Principles and Assumptions

The Messaging Platform is based on the following set of architectural principles:

1. There shall be a common transport layer for all messages. For the NHIN to be a truly scalable, secure and interoperable network, a common transport layer is essential. The Messaging Platform will be based on SOAP 1.2 messages over HTTP. All higher-level messaging elements and attachments must be bound to a SOAP message. This excludes the use of incompatible transport protocols such as HL7 MLLP for NHIO to NHIO messaging.

2. Reliable messaging should be available to support specific information exchanges, but it is not a requirement for every NHIN service.
3. Messages between NHIOs must be secure. This will necessitate the use of encryption as part of the message transport layer.
4. The common message envelope must support assertions about security and trust between NHIOs.
5. PKI is used to establish a technical "trust fabric" for the NHIN
6. The basis for authentication for NHIO participants shall be X.509 certificates. All NHIN certificates will be issued by a common trusted certificate authority. All NHIO to NHIO messages must be digitally signed for purposes of authentication and non-repudiation.
7. The NHIN shall be based on interoperability profiles that have been fully approved as an industry interoperability standard and are capable of being implemented by NHIN nodes using available SOA platforms.
8. In certain well-defined cases, an NHIO may assert specific policy constraints with respect to the invocation of their web services. These may include authentication requirements, encryption requirements, NHIN profiles supported, specific versions of services supported, and other constraint types defined for the NHIN.

2.3 Triggers

The NHIN Messaging Platform is central to the messaging, security, and privacy foundation. All NHIN requests must conform to this specification.

2.4 Transaction Standard

The NHIN Messaging Platform is based on the use of web services, as articulated within interoperability profiles established by the Web Services Interoperability Organization (WS-I). Collectively, the WS-I Basic and Basic Security Profiles define a common platform for secure and reliable exchange of messages between NHIOs in a manner that is independent of specific operating system or programming language frameworks.

The Messaging Platform specification utilizes a single version of all relevant standards to ensure interoperability and consistency, although these versions may be revisited at a later point.

2.4.1 Basic Profile:

WS-I Basic Profile 2.0
http://www.ws-i.org/Profiles/BasicProfile-2_0(WGD).html

Specification	Version	Notes
Simple Object Access Protocol (SOAP)	1.2	
SOAP Message Encoding Style		Document Literal
SOAP Faults		No custom SOAP faults are required.
Hypertext Transfer Protocol (HTTP)	1.1	
WS-Addressing	1.0	
WS-BaseNotification	1.3	
Message Transmission Optimization Mechanism (MTOM) binding for SOAP	1.0	
Web Services Description Language (WSDL)	1.1	
WS – Reliable Messaging	1.2	Use of WS-RM will be constrained to specific NHIN profiles and is not intended for use as part of every NHIN message. Each NHIN profile requiring reliable messaging shall indicate the specific delivery assurance requirements (at least/exactly/at most once, and whether a requirement for ordered delivery exists). These assurance requirements must be expressed using WS-RM Policy assertions. Use of WS-RM requires the "UsesSequenceSSL" attribute as described in section 6.2 of the WS-Reliable Messaging version 1.2 specification, in order to avoid sequence spoofing attacks. The corresponding WS-I profile for reliable messaging, *Reliable Secure Profile 1.0*, will be adopted by the NHIN in the future once the profile has been finalized and achieved industry adoption.
WS-Policy	1.5	Required for WS-RM Policy assertions described above, as well as other policy assertions to be described separately
WS-Policy Attachments	1.2	Defines two general-purpose mechanisms for associating policies with the subjects to which they apply. This specification also defines how these general-purpose mechanisms may be used to associate WS-Policy with WSDL and UDDI descriptions.
WS-Policy Framework	1.2	Provides a general purpose model and corresponding syntax to describe the policies of a Web Service.
WS-Security Policy	1.2	Defines a set of security policy assertions for use with the WS-Policy framework with respect to security features provided in WSS: SOAP Message Security.
XML Schema	1.0	
Universal Discovery and Description Interface (UDDI)	3.0.2	The use of UDDI is fully described in the "NHIN Web Services Registry" specification.

2.4.2 Security Profile:

WS-I Security Profile 1.1
http://www.ws-i.org/Profiles/BasicSecurityProfile-1.1.html

Specification	Version	Notes
Transport Layer Security	1.0	Note that this is equivalent to Secure Sockets Layer 3.0
XML Signature	1.0	
Web Services Description Language (WSDL)	1.1	
Symmetric Encryption Algorithm and Key Length	AES 128-bit	AES = Advanced Encryption Standard
X.509 Token Profile	1.0	
SAML Token Profile	1.1	Note that SAML Token Profile 1.1 mandates the use of the SAML 2.0 Base Specification
Attachment Security	1.0	

3 Transport Definition

3.1 Message Syntax

All messages between NHIN nodes are SOAP messages and therefore implement the SOAP syntax.

3.2 Namespaces

The messages defined in the NHIN Service Interface Specifications utilize elements defined in several different places, as described in the following table. Readers and implementers are urged to pay careful attention to the XML namespaces used in the various NHIN Service Interface Specifications.

Namespace prefix used in this document	Full namespace identifier	Description
wsnt	http://docs.oasis-open.org/wsn/b-2	WS-Base Notification standard
wsa	http://www.w3.org/2005/08/addressing	WS-Addressing standard
nhin	http://www.hhs.gov/healthit/nhin	A NHIN defined namespace
xacml	urn:oasis:names:tc:xacml:2.0:policy:schema:os	XACML standard
rim	urn:oasis:names:tc:ebxml-regrep:xsd:rim:3.0	ebXML Registry Information Model
ihe	urn:ihe:iti:xds-b:2007	IHE XDS.b schema
rs	urn:oasis:names:tc:ebxml-regrep:xsd:rs:3.0	Registry

3.3 Public Key Infrastructure

3.3.1 Certificate Authority

A single NHIN certificate authority (CA) will issue X.509 certificates to all NHIOs. The fact that all certificates are signed by the common designated CA will serve as the basis of trust and authentication between NHIOs; all messages between NHIOs are both signed and encrypted using these certificates.

3.3.2 Certificate Verification and Revocation

A very important CA function is the ability of a node to verify the validity of certificates presented for authentication. In this way, NHIOs can have a high level of assurance that the entities they are

communicating with are indeed who they say they are and are authorized to participate in NHIN communication. Traditionally, there are two ways to verify the validity of a X.509 certificate: 1) via Certificate Revocation Lists (CRL) and 2) through Online Certificate Status Protocol (OCSP). As per the Internet Engineering Task Force's RFC5280 dated May 2008, "CAs are responsible for indicating the revocation status of the certificates that they issue. Revocation status information may be provided using the Online Certificate Status Protocol (OCSP) [RFC2560], certificate revocation lists (CRLs), or some other mechanism."

The following text further constrains the use of certificate revocation checking with respect to RFC5280 and RFC2560 to remove ambiguity.

The NHIN governance body contracts with a Certification Authority (CA) which offers Online Certificate Status Protocol (OCSP) responder services and support the publication of Certificate Revocation Lists (CRLs).

Each initiating and responding NHIO gateway MUST implement either OCSP or CRL-based x.509 certificate revocation checking against the NHIN-managed CA, at the gateway level, to determine the revocation status of each certificate as per NHIN policy, or in the absence of such a policy, for each transaction.

3.3.2.1 OCSP

The NHIN-governed CA supports the use of Online Certificate Status Protocol (OCSP) for x.509 certificate revocation status determination as specified in the Internet Engineering Task Force's RFC2560 "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP" dated June 1999.

Each NHIO MAY employ OCSP-based certificate revocation checking. If a given NHIO elects to use OCSP-based certificate revocation detection, then that NHIO's initiating and responding gateways MUST support NHIN policy with respect to latency, frequency of updates, and availability, as contained in the DURSA.

OCSP-based NHIO gateways, SHOULD implement the cryptographic "nonce" extension to OCSP to help prevent replay attacks as per RFC2560 Section 4.4.1. Other extensions to OCSP, such as additional CA and certificate revocation status information, MAY BE implemented, subject to support by the CA and NHIO implementations. The NHIN-governed CA supports the OCSP HTTP transport protocol GET and POST methods, as per RFC2560 Appendix A.1 and as well as commonly implemented protocols such as LDAP, TFTP, SOAP, and HTTPS.

The NHIN-governed CA supports the use of the Authority Information Access extension as per RFC5280 Section 4.2.2.1 to specify the OCSP Responder end point and access protocol. The certificate id-ad-ocsp accessMethod OID is used to identify that OCSP support is available for each certificate issued by the NHIN-governed CA. The certificate accessLocation specifies the OCSP Responder end point.

Prior to initiating a request to the NHIN-governed CA OCSP Responder, each OCSP-based NHIO gateway MUST confirm the validity of the candidate x.509 certificate (it is within its validity period, the signature is correct, the certificate is not corrupted, etc.) since base implementation OCSP Responders MUST only be used to query the revocation status of valid certificates (OCSP Responders may return unexpected responses for invalid certificates).

3.3.2.2 CRL

The NHIN-governed CA supports the use of Certificate Revocation Lists (CRLs) for x.509 certificate revocation status determination as specified in the Internet Engineering Task Force's RFC5280 "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile" dated May 2008. The NHIN-governed CA supports CRL differential updates (a Delta CRL Distribution Point) in the issued certificate's cRLDistributionPoints extension mechanism as specified in RFC5280 section 4.2.1.15.

If OCSP is not implemented by a given NHIO, the NHIO MUST implement CRL-based certificate revocation checking. If a given NHIO elects to use CRL-based certificate revocation detection, then that NHIO's initiating and responding gateways MUST support NHIN policy with respect to latency, frequency of updates, and availability, as contained in the DURSA.

The NHIN-governed CA supports the CRL and CRL Extensions Profile as per RFC5280 Section 5. The NHIN-governed CA issues "a full and complete list of all unexpired certificates issued by this CA that have been revoked for any reason". This follows the FPKI certificate and CRL profile which does not requre delta CRL's (http://www.idmanagement.gov/fpkipa/documents/fpki_certificate_profile.pdf). The CRL fields are to be implemented as per RFC5280 Section 5.1

3.3.3 Certificates

The NHIN-governed CA will not issue x.509 certificates with a notAfter validity date ASN.1 GeneralizedTime value of 99991231235959Z. It will issue x.509 certificates for a validity period as determined by NHIN policy, or in the absence of such policy, for a validity period of no longer than 12 months. The NHIN-governed CA will also ensure that it issues unique certificate serial numbers.

3.4 Digitally Signing SAML Assertions

The following information is provided as guidance for digitally signing SAML Assertions, which are described in the NHIN Authorization Framework specification.

3.4.1 Policy Definition

Within the WSDL defining an available Web Service, the recipient of the HTTP request has opportunity to assert the various WS-SP policies that define the security expectations that control access to the requested service. It is within this policy that the use of a secure transport is described as well as the use of a selected algorithm suite for use in the digital signature and for encryption purposes. In addition a supporting token is defined that distinguishes the various types of SAML Token Authentication.

Mutual authentication, TLS, is supported through the definition of the sp:TransportBinding policy assertion. The following sp:TransportToken declares that the secure transport will be over Https via the sp:HttpsToken assertion, and it declares mutual authentication via the sp:RequireClientCertificate assertion.

```
<sp:TransportToken>
   <wsp:Policy>
      <sp:HttpsToken>
         <wsp:Policy>
            <sp:RequireClientCertificate/>
         </wsp:Policy>
      </sp:HttpsToken>
   </wsp:Policy>
</sp:TransportToken>
```

The cipher suite recommendation from the OASIS committee for SAML2.0 over the TLS protocol was for forward looking applications to implement TLS_RSA_WITH_AES_128_CBC_SHA, which has both speed and security strength.[1] This translates to the Basic128 algorithm suite as defined in the WS-SecurityPolicy[2] and is declared within the sp:AlgorithmSuite assertion as shown in the following example.

```
<sp:AlgorithmSuite>
   <wsp:Policy>
      <sp:Basic128/>
   </wsp:Policy>
</sp:AlgorithmSuite>
```

[1] Security and Privacy Considerations for the OASIS Security Assertion Markup Language (SAML) V2.0

[2] WS-SecurityPolicy 1.3

The SAML2.0 Holder-of-Key Assertion is declared differently dependent upon the defined binding. In the event of Transport Binding which declares the TLS case, the SAML2.0 Holder-of-Key appears as an sp:EndorsingSupportingToken. Endorsing tokens can also be used to define additional message parts to sign and/or encrypt; however, these are not used at this time. When transport security is defined as in this case, the signature must also cover the message timestamp. The sp:SamlToken distinguishes that this is a SAML Token assertion that will always expect the token to be included on messages sent to the recipient. It is the WssSamlV20Token11 that actually declares this as Saml version 2.0. The following example provides an sp:EndorsingSupportingToken.

```
<sp:EndorsingSupportingTokens>
   <wsp:Policy>
      <sp:SamlToken sp:IncludeToken = " http://docs.oasis-open.org/ws-sx/ws-
securitypolicy/200702/IncludeToken/AlwaysToRecipient">
         <wsp:Policy>
            <sp:WssSamlV20Token11/>
         </wsp:Policy>
      </sp:SamlToken>
   </wsp:Policy>
</sp:EndorsingSupportingTokens>
```

Putting this all together we have the policy requirements to support the declaration of a SAML2.0 assertion over TLS.[3]

```
<sp:TransportBinding>
      <wsp:Policy>
         <sp:TransportToken>
            <wsp:Policy>
               <sp:HttpsToken>
                  <wsp:Policy>
                     <sp:RequireClientCertificate/>
                  </wsp:Policy>
               </sp:HttpsToken>
            </wsp:Policy>
         </sp:TransportToken>
         <sp:Layout>
            <wsp:Policy>
               <sp:Strict/>
            </wsp:Policy>
         </sp:Layout>
         <sp:IncludeTimestamp/>
         <sp:AlgorithmSuite>
            <wsp:Policy>
               <sp:Basic128/>
            </wsp:Policy>
         </sp:AlgorithmSuite>
      </wsp:Policy>
   </sp:TransportBinding>
   <sp:EndorsingSupportingTokens>
      <wsp:Policy>
         <sp:SamlToken sp:IncludeToken = " http://docs.oasis-open.org/ws-sx/ws-
securitypolicy/200702/IncludeToken/AlwaysToRecipient">
            <wsp:Policy>
               <sp:WssSamlV20Token11/>
            </wsp:Policy>
         </sp:SamlToken>
      </wsp:Policy>
   </sp:EndorsingSupportingTokens>
```

[3] WS-SecurityPolicy Examples

3.4.2 Digital Signature Specification

XML Digital Signatures are applied to data objects through an indirection or URI reference. The data object is digested and that digest value is placed within the signature's <ds:DigestValue> element which is cryptographically signed. The table below discusses the elements and attributes for the <ds:Signature> element.

Element/Attribute	Usage	Description
@Id	Optional	Identifies this signature for possible later reference.
SignedInfo	Required	Contains the definition of the Canonicalization method, the Signature method, and the reference to the object being signed.
SignatureValue	Required	Contains the actual value of the digital signature
KeyInfo	Optional (Required for NHIN)	Contains the information such that the recipient can validate the signature. If this is omitted then the recipient is expected to be able to identify the key by some other means. For the current NHIN set-up this should be used to convey this information.
Object ID	Optional	Optional element that may contain other data such as MIME type, an ID, or Encoding attributes.

The <ds:SignedInfo> element is rather like a container for the following elements and attributes.

Element/Attribute	Usage	Description
@Id	Optional	Identifies this element for possible later reference
Canonicalization Method	Required	Must support the required canonicalization algorithms. It is recommended that Exclusive Canonicalization be used.[4] http://www.w3.org/2001/10/xml-exc-c14n#
SignatureMethod	Required	Identifies the cryptographic functions involved in the signature operation. It is recommended that SAML processors support the use of RSA signing and verification.[4] http://www.w3.org/2000/09/xmldsig#rsa-sha1
Reference	Required	This element must contain the URI of that which is being signed. For example this would contain the ID of the Assertion element when signing that element, or the ID of the <wsu:Timestamp> when signing that element.

The <ds:Reference> element specifies the digest algorithm the digest method and what is being signed. For SAML2.0 there is this single contained <ds:Reference> element. It has the following elements and attributes.

[4] Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0

Element/Attribute	Usage	Description
@Id	Optional	Identifies this element for possible later reference.
@URI	Optional (Required for SAML2.0)	For SAML2.0 this must identify the object being signed using that elements Id.[4]
@Type	Optional	Identifies the type of object being signed.
Transforms	Optional	This is an ordered list of transformations that took place on the data object. This is only allowed to contain a subset of enveloped signature transform, exclusive canonicalization transform, and exclusive canonicalization with comments.[4] http://www.w3.org/2000/09/xmldsig# (enveloped-signature) http://www.w3.org/2001/10/xml-exc-c14n# http://www.w3.org/2001/10/xml-exc-c14n# (With Comments).
DigestMethod	Required	Defines the digest algorithm that is applied. For the Basic128 Algorithm Suite this is SHA1. http://www.w3.org/2000/09/xmldsig#sha1
DigestValue	Required	This is the encoded value of the digest.

The <ds:KeyInfo> element provides the means by which the signature is validated. This document will present one possible simple implementation for validating the signature of the Assertion Token by defining only the <ds:KeyValue> element. The KeyValue element contains a single public key that can be used to validate the signature. These are structured formats for defining the DSA (Digital Signature Algorithm) with a recommendation to use the RSA. The <ds:RSAKeyValue> has the following elements:

Element/Attribute	Usage	Description
Modulus	Required	This is a prime modulus used in the DSA.
Exponent	Required	This is the exponent term.

These arbitrary-length integers are represented as octet strings as defined by the ds:CryptoBinary type which is a base64Binary.

The following is an example of the digital signature given the above requirements.

```
<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
   <ds:SignedInfo>
      <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
      <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
      <ds:Reference URI="#51cb7689-0957-46a2-938e-1add75577ab7">
         <ds:Transforms>
            <ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
            <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
         </ds:Transforms>
         <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
         <ds:DigestValue>a3XVN23H2N/ga+08AGqGHD1euKc=</ds:DigestValue>
      </ds:Reference>
   </ds:SignedInfo>
   <ds:SignatureValue>L8Liyz+6pLwNP9YBfIRbrDVUJtM2YcLuN3+HPjspQEHmZ2uTXWYuy7XTM9dqmN93w0ypVM7egjRe=</ds:
SignatureValue>
   <ds:KeyInfo>
      <ds:KeyValue>
         <ds:RSAKeyValue>
            <ds:Modulus>vYxVZKIzVdGMSBkW4bYnV80MV/RgQKV1bf/DoMTX8laMO45P6=</ds:Modulus>
            <ds:Exponent>AQAB</ds:Exponent>
         </ds:RSAKeyValue>
      </ds:KeyValue>
   </ds:KeyInfo>
</ds:Signature>
```

When validating the signing of the timestamp, the <ds:KeyInfo> element will contain the <wsse:SecurityTokenReference>[5] as shown in this following example:

```
<ds:KeyInfo>
   <wsse:SecurityTokenReference wsse11:TokenType="http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0">
      <wsse:KeyIdentifier ValueType="http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLID">51cb7689-0957-46a2-938e-1add75577ab7</wsse:KeyIdentifier>
   </wsse:SecurityTokenReference>
</ds:KeyInfo>
```

The following reference provides more information on the syntax and processing of digital signatures:
W3C XML Signature Syntax and Processing

3.4.3 Subject Confirmation

As part of the validation and processing of the assertion, the receiver must establish the relationship between the subject and claims of the SAML statements and the entity providing the evidence to satisfy the confirmation method defined for the statement. Statements attested for by the holder-of-key method must be associated with one or more holder-of-key SubjectConfirmation elements. The SubjectConfirmation elements must include a <ds:KeyInfo> element that identifies a public key that can be used to confirm the identity of the subject.[6] The SubjectConfirmation element has the following elements and attributes.

Element/Attribute	Usage	Description
@Method	Required	A URI reference that identifies a protocol or mechanism to be used to confirm the subject. For Holder-of-Key this is: urn:oasis:names:tc:SAML:2.0:cm:holder-of-key
BaseId NameId EncryptedId	Optional	Identifies the entity expected to satisfy the enclosing subject confirmation requirements.
SubjectConfirmationData	Optional	However, as this contains the KeyInfo element it is required for Holder-of-Key.

The SubjectConfirmationData element specifies additional data that allows the subject to be confirmed. This can include a variety of optional attributes: NotBefore, NotOnOrAfter, Recipient, InResponseTo, and Address. However, it is the <ds:KeyInfo> element that identifies the cryptographic key that is used to authenticate the attesting entity.[7]

The following is an example of the subject confirmation element given the above requirements.

[5] WS-SecurityPolicy – Appendix C

[6] Web Services Security: SAML Token Profile 1.1

[7] Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0

```
<saml2:Subject>
   <saml2:NameID
     Format="urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName">
      CN=Alex G. Bell,O=1.22.333.4444,UID=abell
   </saml2:NameID>
   <saml2:SubjectConfirmation
    Method="urn:oasis:names:tc:SAML:2.0:cm:holder-of-key">
      <saml2:SubjectConfirmationData>
         <ds:KeyInfo>
            <ds:KeyValue>
               <ds:RSAKeyValue>
                  <ds:Modulus>vYxVZKIzVdGMSBkW4bYnV80MV/RgQKV1bf/D
                    X8laMO45P6rlEarxQiOYrgzuYp+snzz2XM0S6o3JGQtXQ=
                     <ds:Modulus>
                  <ds:Exponent>AQAB</ds:Exponent>
               </ds:RSAKeyValue>
            </ds:KeyValue>
         </ds:KeyInfo>
      </saml2:SubjectConfirmationData>
   </saml2:SubjectConfirmation>
</saml2:Subject>
```

4 Error Handling

No custom SOAP faults are required by this specification. Appendix A contains a sample SOAP Fault Message.

Each of the NHIN Specifications contains a section on error handling for describing the constraints and extensions on the base specifications they use for faults. The reader is directed to each particular specification for those details.

5 Auditing

When an NHIO should create an “Export” audit event or an “Import” audit event when sending or receiving messages to another NHIO is detailed in the various service interface specifications and the reader is directed to the particular specification for those details.

Appendix A: Sample SOAP Messages

Sample SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope"
xmlns:wsa="http://www.w3.org/2005/08/addressing" xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd" xmlns:wsu="http://docs.oasis-open.org//wss/2004/01/oasis-200401-wsswssecurity-utility-1.0.xsd"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
  <soapenv:Header>
  <!--  MessageID, To and Action are required elements  -->
  <!--  Unique identifier of this message   -->
  <wsa:MessageID>urn:uuid:0fbfdced-6c01-4d09-a110-2201afedaa02</wsa:MessageID>
  <!--  URI of the service    -->
  <wsa:To soapenv:mustUnderstand="1">http://stelsewhere.com/XdsService/IHEXDSRepository.svc</wsa:To>
  <!--  URI indicates specific action that is requested to be performed by the service   -->
  <!--  Same as To URI in HTTP requests   -->
  <!--  In a non-HTTP request, To and Action may be different, with Action pointing to the WSDL PortType    -->
  <wsa:Action soapenv:mustUnderstand="1">urn:ihe:iti:2007:RetrieveDocumentSet</wsa:Action>
  <!--  URI of the requester   -->
  <wsa:From>http://http://generalhospital.org/nhiegateway/getDocs</wsa:From>
  <!--  URI to which the response should be sent    -->
  <!--  If ReplyTo is not the same as From, treat as an Asynchronous req/response   -->
  <!--  If ReplyTo is the same as From, or is Anonymous, or omitted treat as a Synchronous request   -->
  <wsa:ReplyTo>
  <wsa:Address>http://generalhospital.org/nhiegateway/getDocs</wsa:Address>
  </wsa:ReplyTo>
  <wsse:Security soapenv:mustUnderstand="true">
  <wsse:UsernameToken>
  <!--  For Audit logging purposes, a unique Username from the Requesting Organization / NHIO is also sent   -->
  <wsse:Username>John Doe MD, General Hospital</wsse:Username>
  </wsse:UsernameToken>
  <wsse:BinarySecurityToken wsu:Id="X509Token" ValueType="wsse:X509v3"
EncodingType="wsse:Base64Binary">MIIOlskew78Hjkds...</wsse:BinarySecurityToken>
  <!--  Timestamp of the Security element   -->
  <!--  Often digitally signed to prevent replay attacks   -->
  <wsu:Timestamp wsu:Id="MesgTimeStamp">
  <wsu:Created>2008-03-14T15:42:00Z</wsu:Created>
  <wsu:Expires>2008-03-14T16:00:00Z</wsu:Expires>
  </wsu:Timestamp>

  <!--  Digital Signature used to sign parts of this document   -->
  <ds:Signature>
 <ds:SignedInfo>
  <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
  <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
  <ds:Reference URI="#MesgTimeStamp">
  <ds:Transforms>
  <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
  </ds:Transforms>
  <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
  <ds:DigestValue>LyLsF094hPi4wPU...</ds:DigestValue>
  </ds:Reference>
  <ds:Reference URI="#nhinMesg">
  <ds:Transforms>
  <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
  </ds:Transforms>
  <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
  <ds:DigestValue>LyLsF094hPi4wPU...</ds:DigestValue>
  </ds:Reference>
  </ds:SignedInfo>
  <ds:SignatureValue>Hp1ZkmFZ/2kQLXDJbchm5gK...</ds:SignatureValue>
  <ds:KeyInfo>
  <wsse:SecurityTokenReference>
  <wsse:Reference URI="#X509Token" />
  </wsse:SecurityTokenReference>
  </ds:KeyInfo>
  </ds:Signature>
```

```
 </wsse:Security>
 </soapenv:Header>
 <soapenv:Body>
 <reqMesg wsu:Id="nhinMesg">

 </reqMesg>
 </soapenv:Body>
 </soapenv:Envelope>
```

Sample SOAP Response

```
<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope"
xmlns:wsa="http://www.w3.org/2005/08/addressing">
 <soapenv:Header>
 <!-- MessageID, To and Action are required elements -->
 <!-- Unique identifier of this message -->
 <wsa:MessageID>urn:uuid:f0dedced-6c01-4d09-a110-2201afedf0f0</wsa:MessageID>
 <!-- Requester URI -->
 <wsa:To soapenv:mustUnderstand="1">http://http://generalhospital.org/nhiegateway</wsa:To>
 <!-- Same as To URI in HTTP requests -->
 <!-- In a non-HTTP request, To and Action may be different, with Action pointing to the WSDL PortType -->
 <wsa:Action oapenv:mustUnderstand="1">urn:ihe:iti:2007:RetrieveDocumentSetResponse</wsa:Action>
 <!-- Unique identifier of the original message to which this is a response -->
 <wsa:RelatesTo>urn:uuid:0fbfdced-6c01-4d09-a110-2201afedaa02</wsa:RelatesTo>
 </soapenv:Header>
 <soapenv:Body>
 <respMesg>RESPONSE Document ...</respMesg>
 </soapenv:Body>
 </soapenv:Envelope>
```

Sample SOAP Fault

```
<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope/"
xmlns:wsa="http://www.w3.org/2005/08/addressing">
 <soapenv:Header>
 <!-- MessageID, To and Action are required elements -->
 <!-- Unique identifier of this message -->
 <wsa:MessageID>urn:uuid:f0dedced-6c01-4d09-a110-2201afedf0f0</wsa:MessageID>
 <!-- Requester URI -->
 <wsa:To soapenv:mustUnderstand="1">http://http://generalhospital.org/nhiegateway</wsa:To>
 <!-- Same as To URI in HTTP requests -->
 <!-- In a non-HTTP request, To and Action may be different, with Action pointing to the WSDL PortType -->
 <wsa:Action soapenv:mustUnderstand="1">urn:ihe:iti:2007:RetrieveDocumentFault</wsa:Action>
 <!-- Unique identifier of the original message to which this is a Fault response -->
 <wsa:RelatesTo>urn:uuid:0fbfdced-6c01-4d09-a110-2201afedaa02</wsa:RelatesTo>
 </soapenv:Header>
 <soapenv:Body>
 <soapenv:Fault>
 <soapenv:Code>
 <soapenv:Value>env:Client</soapenv:Value>
 </soapenv:Code>
 <soapenv:Reason>
 <soapenv:Text xml:lang="en">There was an error in the incoming SOAP request</soapenv:Text>
 </soapenv:Reason>
 </soapenv:Fault>
 </soapenv:Body>
 </soapenv:Envelope>
```