NEMO Technology Platform Specifications

Version 1.1.1
Final

Source	Marlin Developer Community
Date	February 5, 2010

Notice

THIS DOCUMENT IS PROVIDED "AS IS" WITH NO REPRESENTATION OR WARRANTY, EXPRESS OR IMPLIED, CONCERNING THE COMPLETENESS, ACCURACY, OR APPLICABILITY OF ANY INFORMATION CONTAINED IN THIS DOCUMENT. THE MARLIN DEVELOPER COMMUNITY (“MDC”) ON BEHALF OF ITSELF AND ITS PARTICIPANTS (COLLECTIVELY, THE "PARTIES") DISCLAIM ALL LIABILITY OF ANY KIND WHATSOEVER, EXPRESS OR IMPLIED, ARISING OR RESULTING FROM THE RELIANCE OR USE BY ANY PARTY OF THIS DOCUMENT OR ANY INFORMATION CONTAINED HEREIN. THE PARTIES COLLECTIVELY AND INDIVIDUALLY MAKE NO REPRESENTATIONS CONCERNING THE APPLICABILITY OF ANY PATENT, COPYRIGHT (OTHER THAN THE COPYRIGHT TO THE DOCUMENT DESCRIBED BELOW) OR OTHER PROPRIETARY RIGHT OF THIS DOCUMENT OR ITS USE, AND THE RECEIPT OR ANY USE OF THIS DOCUMENT OR ITS CONTENTS DOES NOT IN ANY WAY CREATE BY IMPLICATION, ESTOPPEL OR OTHERWISE, ANY LICENSE OR RIGHT TO OR UNDER ANY PATENT, COPYRIGHT, TRADEMARK OR TRADE SECRET RIGHTS WHICH ARE OR MAY BE ASSOCIATED WITH THE IDEAS, TECHNIQUES, CONCEPTS OR EXPRESSIONS CONTAINED HEREIN.

Use of this document is subject to the agreement executed between you and the Parties, if any.

Any copyright notices shall not be removed, varied, or denigrated in any manner.

Intellectual Property

A commercial implementation of this specification requires a license from the Marlin Trust Management Organization.

Contact Information

Feedback on this specification should be addressed to: editor@marlin-community.com

Contact information for the Marlin Trust Management Organization can be found at: http://www.marlin-trust.com/

Table of Contents

1 Introduction

This document contains the NEMO technology platform specifications.

Please note: This document consolidates material that was previously in separate documents. The highest version number among the source documents consolidated into this document was 1.1. The version number for this document is initially one greater than that in the third digit, that is, 1.1.1.

The NEMO (Networked Environment for Media Orchestration) framework provides the trusted "plumbing" between the various functional components in a system. NEMO combines SOAP web services with SAML authorizations to provide end-to-end message integrity and confidentiality protection, entity authentication, and role-based service authorization. Through the use of the NEMO framework, components can leverage a consistent mechanism to ensure that messages are delivered with appropriate protection and are exchanged between entities that are properly authenticated and authorized.

There are five NEMO specifications, all of which are provided in this document:

1. NEMO Message Bindings (§2)

   This specifies XML-related bindings pertaining to describing and communicating with NEMO services.

2. NEMO Security Bindings (§3)

   This describes how to implement a set of NEMO Secure Messaging Protocols using a subset of OASIS' Web Services Security standard and related documents.

3. NEMO Trust Management Bindings (§4)

   This describes bindings of NEMO trust management mechanisms—in particular, the use of SAML-specified URIs for NEMO node identifiers, the use of X.509 certificates for NEMO node authentication, the use of SAML attribute assertions with NEMO nodes, and the definition of a special NEMO node "role" attribute.

4. NEMO Policy Bindings (§5)

   This specifies bindings that can be used to express policies defining the security requirements for the NEMO Secure Messaging Protocol bindings.

5. NEMO Discovery/Inspection Bindings (§6)

   This specifies XML-related bindings pertaining to NEMO Inspection and Discovery. Discovery is the ability to search for services offered by NEMO nodes based on different criteria and to obtain references to where those services can be bound to for access. Inspection is the ability to query a given NEMO node reference about certain well-defined attributes (metadata) in regards to its state, such as descriptions of the policy related to the services it publicly offers.

The final section of this document, §7, provides a table with complete references for the external documents referred to within this document.

1.1 Namespaces

The following namespaces are used in this document:

Prefix	Namespace
ds	http://www.w3.org/2000/09/xmldsig#
enc	http://www.w3.org/2001/04/xmlenc#
nemo	http://nemo.intertrust.com/2004
nemoc	http://nemo.intertrust.com/2005/10/core
nemop	http://nemo.intertrust.com/2004/policy
nemosec	http://nemo.intertrust.com/2005/10/security
S11	http://schemas.xmlsoap.org/soap/envelope
S12	http://www.w3.org/2003/05/soap-envelope
saml	urn:oasis:names:tc:SAML:1.0:assertion
soap	http://schemas.xmlsoap.org/soap/envelope/
wsa	http://www.w3.org/2005/08/addressing
wsc	http://schemas.xmlsoap.org/ws/2004/04/sc
wsd	http://schemas.xmlsoap.org/ws/2004/10/discovery
wsdl	http://schemas.xmlsoap.org/wsdl/
wsp	http://schemas.xmlsoap.org/ws/2002/12/policy
wsse	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd
wssp	http://schemas.xmlsoap.org/ws/2002/12/secext/
wst	http://schemas.xmlsoap.org/ws/2004/04/trust
wsu	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd
xs	http://www.w3.org/2001/XMLSchema
xsd	http://www.w3.org/2001/XMLSchema

1.2 Notation

The entities `&nemo;`, `&nemoc;` `&nemop`;, and `&nemosec;` are defined to provide shorthand identifiers for URIs defined in this specification. For example,

```
&nemo;/Element
```

corresponds to

```
http://nemo.intertrust.com/2004/Element
```

and

```
&nemop;/Element
```

corresponds to

```
http://nemo.intertrust.com/2004/policy/Element
```

These entities are used as shorthand notation throughout this document.

2 NEMO Message Bindings

2.1 Overview

This section specifies XML-related bindings pertaining to describing and communicating with NEMO services.

2.2 SOAP Binding

SOAP ("Simple Object Access Protocol") is a lightweight wire protocol used to package one-way messages in a distributed environment. It forms the basis for more complex messaging patterns and is the standard protocol for invoking web services. SOAP message headers provide a mechanism for supporting session management and communicating security information within NEMO.

NEMO messages SHALL comply with [SOAP 1.1] and with the Web Services-Interoperability Organization Basic Profile 1.1 [WSIBasicProfile11] and Simple SOAP Binding Profile [WSISOAPBinding].

2.2.1 FaultDetails Header Element

According to [SOAP 1.1], details of faults generated by processing header elements MUST be communicated in the header, rather than the body, of a subsequent message. For this purpose, NEMO nodes MAY use the `<nemoc:FaultDetails>` element as an immediate child of the `<S11:Header>` element of the subsequent message.

2.2.2 wsa:Action and soapAction

Every NEMO SOAP message document MUST contain a `<wsa:Action>` element compliant with [WS-ADDR].

2.3 Message Correlation

Message correlation may be required by application logic to implement stateful message exchange patterns between NEMO nodes. This section describes three usage patterns for correlating messages using mechanisms from WS-Addressing [WS-ADDR]. In each usage pattern, correlated messages MUST contain message information headers compliant with [WS-ADDR].

This specification defines a WS-Addressing relationship type:

URI	Description
`&nemo;/addressing/originatesFrom`	Indicates that this message follows (perhaps indirectly) from the given original message.

A message MUST NOT originate from more than one message. Define a message to be *original* if it originates from itself. If a message originates from a second message, the second message MUST be original.

Note: These rules make it possible to identify a group of correlated messages by the message ID of the original message.

Note: It is possible for a message to be in reply to (i.e., have a relationship type of *reply*, as in [WS-ADDR]), with more than one message.

Three OPTIONAL correlation usage patterns are described below.

1. Messages are correlated by the presence of `<wsa:RelatesTo>` elements containing either no `@RelationshipType` attribute or `@RelationshipType` attributes equal to the URI specified in [WS-ADDR] for message replies. A message is correlated transitively, reflexively and symmetrically with all messages made in reply and messages replied to. In particular, two replies to the same message are correlated.
2. Messages are correlated by the presence of `<wsa:RelatesTo>` elements containing `@RelationshipType` attributes equal to the URI specified above for a message that originates from an original message. A message is correlated with all messages that originate from the same original message.
3. Messages are correlated by both of the previous mechanisms. The two mechanisms MUST define the same set of correlated messages.

Note (non-normative): These message correlation patterns may be applied to particular message exchange patterns, such as a linearly ordered sequence of messages between two NEMO nodes. In a linearly ordered sequence, the WS-Addressing reply mechanism provides message sequencing as well as message correlation.

2.3.1 Example

The following three messages form a complete request-response-confirm message exchange between two NEMO nodes. The messages are correlated using both mechanisms defined in the previous section.

```
<!-- Request message -->
<soap:Envelope
   xmlns:soap=”http://www.w3.org/2003/05/soap-envelope”
   xmlns:wsa=”http://schemas.xmlsoap.org/ws/2004/08/addressing”
   xmlns:nemoc=”http://nemo.intertrust.com/2004/core”
   xmlns:tns=”http://example.com/myNamespace”>
   <soap:Header>
      <wsa:To>http://example.com/myService</wsa:To>
      <wsa:Action>http:://example.com/myService/myOperation</wsa:Action>
      <wsa:ReplyTo>
         <wsa:Address>http://fabrikam.com/client</wsa:Address>
      </wsa:ReplyTo>
      <wsa:MessageID>uuid:aaaabbbb-cccc-0001</wsa:MessageID>
      <wsa:RelatesTo
RelationshipType=”http://nemo.intertrust.com/2004/addressing/originatesF
rom”> uuid:aaaabbbb-cccc-0001</wsa:RelatesTo>
   </soap:Header>
   <soap:Body/>
</soap:Envelope>
```

```
<!-- Response message -->
<soap:Envelope
   xmlns:soap=”http://www.w3.org/2003/05/soap-envelope”
   xmlns:wsa=”http://schemas.xmlsoap.org/ws/2004/08/addressing”
   xmlns:nemoc=”http://nemo.intertrust.com/2004/core”
   xmlns:tns=”http://example.com/myNamespace”>
   <soap:Header>
      <wsa:To>http://fabrikam.com/client</wsa:To>
      <wsa:Action>http:://example.com/myNamespace/myPortType/
myOperationResponse</wsa:Action>
      <wsa:ReplyTo>
         <wsa:Address>http://example.com/myService</wsa:Address>
      </wsa:ReplyTo>
      <wsa:MessageID>uuid:aaaabbbb-cccc-0002</wsa:MessageID>
      <wsa:RelatesTo
RelationshipType=”http://nemo.intertrust.com/2004/addressing/originatesF
rom”> uuid:aaaabbbb-cccc-0001</wsa:RelatesTo>
      <wsa:RelatesTo
RelationshipType=”http://www.w3.org/2005/08/addressing/reply”>uuid:aaaab
bbb-cccc-0001</wsa:RelatesTo>
   </soap:Header>
   <soap:Body/>
</soap:Envelope>
```

```
<!-- Confirmation message -->
<soap:Envelope
   xmlns:soap=”http://www.w3.org/2003/05/soap-envelope”
   xmlns:wsa=”http://schemas.xmlsoap.org/ws/2004/08/addressing”
   xmlns:nemoc=”http://nemo.intertrust.com/2004/core”
   xmlns:tns=”http://example.com/myNamespace”>
   <soap:Header>
      <wsa:To>http://example.com/myService</wsa:To>
      <wsa:Action>http:://example.com/myNamespace/myPortType/
myOperationConfirmation</wsa:Action>
      <wsa:MessageID>uuid:aaaabbbb-cccc-0003</wsa:MessageID>
      <wsa:RelatesTo
RelationshipType=”http://nemo.intertrust.com/2004/addressing/originatesF
rom”> uuid:aaaabbbb-cccc-0001</wsa:RelatesTo>
      <wsa:RelatesTo
RelationshipType=”http://www.w3.org/2005/08/addressing/reply”>uuid:aaaab
bbb-cccc-0002</wsa:RelatesTo>
   </soap:Header>
   <soap:Body/>
</soap:Envelope>
```

2.4 *Service Description Bindings*

WSDL is an XML-based language for describing Web services. Service consumers access WSDL documents to inspect the service interfaces and bindings—in this context, this means a binding to a protocol for communicating with the service—and to locate the service endpoint, that is, the network addressable location for submitting the service invocation request message.

NEMO services SHALL be described using service descriptions as specified in [WSDL 1.1].

2.4.1 SOAP Header Faults

Where applicable, the `nemoc:FaultDetails` qualified name SHOULD be specified within a `<wsdl:headerfault>` element, as specified in [WSDL 1.1].

3 NEMO Security Bindings

3.1 *WS-Security Binding*

3.1.1 Overview

The WS-Security binding specified in this section describes how to implement a set of NEMO Secure Messaging protocols, using a subset of OASIS' Web Services Security standard [WS-SEC] and related [WS-*] documents. The specified binding is then applied to define the Basic Secure Messaging Protocol described in §3.1.4.

3.1.2 NEMO Secure Messaging Protocol Elements

The NEMO Secure Messaging Protocol defines how NEMO nodes perform secure message exchange. The following protocol-specific elements are passed with messages to ensure their security:

- Timestamps
- Nonces
- Symmetric keys
- Asymmetric keys
- Encrypted data
- Digital signatures

Web Services Security and related standards define syntax for passing those elements inside SOAP messages. This section provides the most important aspects of this syntax.

3.1.3 Message Security Elements

3.1.3.1 Timestamp

Whenever timestamp support is required, each message MUST contain a `<wsu:Timestamp>` element in the security header. A `<wsu:Timestamp>` element appearing in a `<wsse:Security>` element MUST contain a `<wsu:Created>` element.

3.1.3.2 Nonce

A nonce MAY be carried Base64Encoded in a `<wsse:Nonce>` element.

3.1.3.3 Digital Signature

Whenever message integrity is required, each message MUST contain a digital signature of the appropriate message elements as a `<ds:Signature>` element in the security header. The elements to be signed are specified by this protocol description; other elements MAY also be signed. Detached signatures SHALL be used as specified in [WS-SEC] Section 8.

3.1.3.3.1 Canonicalization Algorithm

The signatures SHALL use exclusive canonicalization, as defined by [EXC-C14N] and recommended by [WS-SEC] and [WSIBasicSecurityProfile10].

3.1.3.3.2 Signing Algorithm

Message integrity for the SOAP Message Binding SHALL be provided with XML Signature [XMLDSIG]. One of the following algorithms SHALL be used [XMLDSIG], [XMLDSIG-MORE].

Purpose	Name	URI
Asymmetric Signature	RSA-SHA256	http://www.w3.org/2001/04/xmldsig-more#rsa-sha256

3.1.3.3.3 Message Digest Algorithm

XML Signatures SHALL digest referenced elements using one of the following algorithms. See [XMLDSIG], [XMLDSIG-MORE], [XML-ENC].

Purpose	Name	URI
Digest within Signature	SHA256	http://www.w3.org/2001/04/xmlenc#sha256

3.1.3.4 Encryption

Whenever message confidentiality is required, each message MUST be encrypted by mechanisms specified in [WS-SEC]. If the key used to encrypt the data needs to be transported, it SHALL be covered with a key encryption key and transported in the SOAP header, as recommended by [WS-SEC] Section 9. If the key used to encrypt the data is a public key belonging to a NEMO node, the embedded `<ds:KeyInfo>` element SHALL identify the public key in a manner consistent with [WS-SECX509] and [WS-SECX509-ERR]. The elements to be encrypted are specified by this protocol description; other elements MAY also be encrypted.

3.1.3.4.1 Encryption Algorithms

Message confidentiality for the SOAP Message Binding SHALL be provided with XML Encryption [XMLENC]. The following algorithms SHALL be used:

Purpose	Name	URI
Data Encryption (Symmetric)	AES-128	http://www.w3.org/2001/04/xmlenc#aes128-cbc
Key Transport (Aysmmetric)	RSA-OAEP	http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p

3.1.3.5 NEMO-Specific Elements

3.1.3.5.1 Usage Attribute

The `nemosec:Usage` attribute SHALL be used to identify a token that has a distinguished role in a certain context, such as a communications protocol or an authorization policy. Alternatively, the `nemosec:TargetUsage` attribute MAY be included in a `<nemosec:Reference>` element that references the token to be identified, as in §3.1.3.5.4. If a token or element within a message has an associated `nemosec:Usage` value, then the element MUST be identified by the attachment of a `nemosec:Usage` attribute to the element or a `nemosec:TargetUsage` attribute to a corresponding `<nemosec:Reference>` element. An element MAY be the target of more than one `<nemosec:Reference>` element.

3.1.3.5.2 Receiver NEMO Node

The element `<nemosec:ToNode>` MAY be used to indicate the intended recipient of a SOAP message. The `<nemosec:ToNode>` element SHOULD appear in the `<S11:Header>`, and MAY appear within a `<wsse:Security>` element.

3.1.3.5.2.1 Syntax

The syntax for `<nemosec:ToNode>` is as follows:

```
<ToNode wsu:Id="…">
   urn:nemo:node:…
</ToNode>
```

…/ToNode

The string content of this element is the canonicalized URI identifier of the intended recipient NEMO node, in UTF-8 encoding, as specified by NEMO Trust Management Bindings in §4.2.

…/ToNode/@{any}

This is an extensibility mechanism to allow additional attributes, based on schemas, to be added.

…/ToNode/{any}

This is an extensibility mechanism to allow different (extensible) ways of identifying NEMO nodes, based on a schema, to be passed. Unrecognized elements MAY cause a fault.

3.1.3.5.3 Sender NEMO Node

The element `<nemosec:FromNode>` MAY appear in the `<S11:Security>` header to indicate the sender of a SOAP message. The `<nemosec:FromNode>` SHOULD be present if the message is not signed. If the message is signed and the `<nemosec:FromNode>` element is present, then the element SHALL be signed.

3.1.3.5.3.1 Syntax

The syntax for `<nemosec:FromNode>` is as follows:

```
   <FromNode wsu:Id=”…”>
      urn:nemo:node:…
   </FromNode>
```

…/FromNode

The string content of this element is the canonicalized URI identifier of the original sending NEMO node, in UTF-8 encoding, as specified by NEMO Trust Management Bindings in §4.2.

…/FromNode/@{any}

This is an extensibility mechanism to allow additional attributes, based on schemas, to be added.

…/FromNode/{any}

This is an extensibility mechanism to allow different (extensible) ways of identifying NEMO nodes, based on a schema, to be passed. Unrecognized elements MAY generate a fault.

3.1.3.5.4 Protocol Declaration

The element `<nemosec:ProtocolDeclaration>` SHALL be used to indicate the use of a named cryptographic protocol. A `<wsse:Security>` element MUST NOT contain more than one `<nemosec:ProtocolDeclaration>` element. If a `<nemosec:ProtocolDeclaration>` element appears in a `<wsse:Security>` element, the `<nemosec:ProtocolDeclaration>` element MUST appear in the `<wsse:Security>` header before all other NEMO-specified elements. If the `<nemosec:ProtocolDeclaration>` element appears in a `<wsse:Security>` element in an `<S11:Header>` element, then the `<wsse:Security>` element and the `<S11:Body>` element MUST contain the elements required for a message of the named protocol. When appearing within a `<wsse:Security>` element, the `<nemosec:ProtocolDeclaration>` element MUST contain a `nemosec:Usage` attribute with value

```
&nemosec;/secure-protocol
```

3.1.3.5.4.1 Syntax

The syntax for `<nemosec:ProtocolDeclaration>` is as follows:

```
   <ProtocolDeclaration wsu:Id=”…” URI=”…” nemosec:Usage=”…”>
      <Step index=”…” Type=”…”/>
      <Reference nemosec:TargetUsage=”…” URI=”…”/>
   </ProtocolDeclaration>
```

…/ProtocolDeclaration

An element identifying a messaging protocol.

…/ProtocolDeclaration/@URI

The URI identifier of the protocol.

…/ProtocolDeclaration/Step

An element identifying a step in a messaging protocol.

…/ProtocolDeclaration/Step/@Index

An optional integer attribute, indicating the sequence number of a message within a protocol. The first message in a protocol SHALL have the number zero.

…/ProtocolDeclaration/Step/@Type

An optional string attribute, indicating the type of message within a protocol. Protocol step types are specific to each protocol. Unrecognized message types MAY generate a fault. If a message is being sent to indicate premature termination of a protocol due to an error condition, the message sender MAY include a `<nemosec:ProtocolDeclaration>` element containing a `<nemosec:Step>` element with a `Type` attribute containing the predefined string

```
fault
```

Other more specific fault step types that MAY be used are defined later in this specification. The general fault step defined here has no implied protocol-related processing rules. Fault step types MUST NOT be used if the security protocol is not abnormally terminated, even if the message body contains an `<S11:Fault>` element. Service access authorization is not part of the NEMO Basic Secure Messaging Protocol (§3.1.4), so authorization failures MUST NOT be reported using fault step types.

…/ProtocolDeclaration/Step/@{any}

This is an extensibility mechanism to allow additional attributes, based on schemas, to be added.

…/ProtocolDeclaration/Reference

Identifies an XML element as filling a distinguished role within the declared security protocol.

…/ProtocolDeclaration/Reference/@nemosec:TargetUsage

Declares the distinguished protocol or policy role or roles that an element fills.

…/ProtocolDeclaration/Reference/@URI

Identifies an XML element that fills a distinguished protocol or policy role or roles. The URI attribute value SHOULD be a shorthand XPointer [XPOINTER].

…/ProtocolDeclaration/@{any}

This is an extensibility mechanism to allow additional attributes, based on schemas, to be added.

…/ProtocolDeclaration/{any}

This is an extensibility mechanism to allow different (extensible) ways of identifying NEMO messaging protocols, based on a schema, to be passed. Unrecognized elements MAY cause a fault.

3.1.3.5.5 Correspondence of WSDL 1.1 Messages and Basic Secure Messaging Protocol Step Types

When the Basic Secure Messaging Protocol (§3.1.4) is used in conjunction with [WSDL 1.1] request-response operations, input messages SHALL be sent with Basic Secure Messaging Protocol request or confirmation step types, and output messages SHALL be sent with Basic Secure Messaging Protocol response messages. Fault messages MAY be sent with either Basic Secure Messaging Protocol response or fault-request step types, depending on whether the fault condition indicates an error within the Basic Secure Messaging Protocol.

3.1.3.5.6 Profile

The element `<nemosec:Profile>` MAY be used to indicate the use of a named NEMO interface profile. If the `<nemosec:Profile>` element appears in a `<wsse:Security>` element in an `<S11:Header>` element, then the containing message MUST conform to the specifications of the indicated NEMO profile. The `<nemosec:Profile>` element MAY contain a `nemosec:Usage` attribute with value

```
&nemosec;/profile
```

3.1.3.5.6.1 Syntax

The syntax for `<nemosec:Profile>` is as follows:

```
<Profile wsu:Id=”…” URI=”…”/>
```

…/Profile

An element defining a NEMO profile.

…/Profile/@URI

The URI identifier of the profile. This specification does not define any profile identifiers.

…/Profile/@{any}

This is an extensibility mechanism to allow additional attributes, based on schemas, to be added.

…/Profile/{any}

This is an extensibility mechanism to allow different (extensible) ways of identifying NEMO profiles, based on a schema, to be passed. Unrecognized elements MAY cause a fault.

3.1.3.5.7 BinarySecurityToken ValueTypes

The following URIs are defined to indicate the `ValueType` of a `<wsse:BinarySecurityToken>`.

Symmetric Key	`&nemosec;/BST/SymmetricKey`

3.1.3.5.8 Role Assertions

The NEMO Trust Management Binding section (§4) defines the use of SAML 1.1 attribute assertions to assert NEMO node roles. According to [WS-SEC-SAML], SAML assertions may be placed within a `<wsse:Security>` element, and may be referenced from a `<wsse:SecurityTokenReference>` element.

A `<wsse:SecurityTokenReference>` element that references a SAML attribute assertion that asserts a NEMO node role MAY contain a `nemosec:Usage` attribute with the value

```
&nemo;/attribute/role
```

3.1.3.5.9 Fault Codes

Messages containing faults MAY use any fault codes defined in [SOAP 1.1] or [WS-SEC]. The following SOAP fault codes MAY also be returned in a `<faultcode>` element within an `<S11:Fault>` element, as specified in [SOAP 1.1]. The fault codes are defined in the `nemosec:` namespace.

Name	Meaning
UnsupportedSecureProtocol	A message contained a `<nemosec:ProtocolDeclaration>` that specifies an unsupported secure protocol.

3.1.4 Basic Secure Messaging Protocol

The Basic Secure Messaging Protocol is a security protocol for NEMO service request and response messages, one-way messages, and three-message (request-response-confirm) message exchange patterns. Basic secure messaging supports optional confidentiality, integrity, and freshness protections.

3.1.4.1 Full Security

The full security version of the basic secure messaging protocol provides confidentiality, integrity, and freshness protection of NEMO service request and response messages. A one-way message exchange protocol includes only a request message; a request-response message exchange pattern includes a request message and a response message; a request-response-confirmation includes a request message, a response message and a confirmation message. The descriptions below apply to all three of these message exchange patterns. There is no signaling within the secure protocol as to which message exchange pattern is in effect.

3.1.4.1.1 Request Message

3.1.4.1.1.1 Security Header

The request SOAP message transmitted from the requestor (client, or service consumer) to the responder (service, or service provider) SHALL contain a `<wsse:Security>` header [WS-SEC], with no actor or role attribute, that contains the following elements: protocol identifier, requestor’s timestamp, requestor’s nonce, responder’s identifier, self-encrypted message key, requestor’s public signing key certificates, requestor’s encrypted one-time message encryption key, and a signature. The security header MAY also contain the requestor’s long-term public message encryption key certificate, SAML assertions [SAML1.1] and a profile identifier.

3.1.4.1.1.1.1 Protocol Identifier

The `<wsse:Security>` element SHALL include a `<nemosec:ProtocolDeclaration>` element with a URI attribute having the value

```
&nemosec;/secure-protocol/basic/1.0
```

The `<nemosec:ProtocolDeclaration>` element SHOULD include a `<nemosec:Step>` element with a `Type` attribute containing the following string:

```
request
```

3.1.4.1.1.1.2 Requestor's Timestamp

The `<wsse:Security>` element SHALL contain a `<wsu:Timestamp>` element that indicates the time at which the requestor sends the request message. The `<wsu:Timestamp>` element MAY contain a `nemosec:Usage` attribute with a value containing the value of the URI attribute in the `<nemosec:ProtocolDeclaration>` with the following fragment concatenated at the end:

```
#request-timestamp
```

3.1.4.1.1.1.3 Requestor's Nonce

The `<wsse:Security>` element SHALL contain a `<wsse:Nonce>` as specified in §3.1.3.2 containing a base64-encoded octet string of no more than 128 octets generated by the requestor. The `<wsse:Nonce>` element MAY contain a `nemosec:Usage` attribute with a value containing the value of the URI attribute in the `<nemosec:ProtocolDeclaration>` with the following fragment concatenated at the end:

```
#request-nonce
```

3.1.4.1.1.1.4 Requestor's Identifier

The `<wsse:Security>` element MAY contain a `<nemosec:FromNode>` element containing the NEMO node ID of the requestor. The `<nemosec:FromNode>` element MAY contain a `nemosec:Usage` attribute with a value containing the value of the URI attribute in the `<nemosec:ProtocolDeclaration>` with the following fragment concatenated at the end:

```
#request-fromNode
```

3.1.4.1.1.1.5 Responder's Identifier

The `<wsse:Security>` element SHALL contain a `<nemosec:ToNode>` element containing the NEMO node ID of the responder. The `<nemosec:ToNode>` element MAY contain a `nemosec:Usage` attribute with a value containing the value of the URI attribute in the `<nemosec:ProtocolDeclaration>` with the following fragment concatenated at the end:

```
#request-toNode
```

3.1.4.1.1.1.6 Self-encrypted Message Key

The `<wsse:Security>` element SHALL contain an `<enc:EncryptedData>` element encrypted with the message encryption key, containing a `<wsse:BinarySecurityToken>`

element containing the cleartext symmetric message key. The `<wsse:BinarySecurityToken>` element SHOULD contain a `ValueType` attribute, as specified in §3.1.3.5.7. The `<wsse:BinarySecurityToken>` element MAY contain a `nemosec:Usage` attribute with a value containing the value of the URI attribute in the `<nemosec:ProtocolDeclaration>` with the following fragment concatenated at the end:

```
#request-messageKey
```

3.1.4.1.1.1.7 Requestor's Public Signing Key Certificates

The `<wsse:Security>` element SHALL contain a `<wsse:BinarySecurityToken>` with a `ValueType` attribute of

```
http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-
profile-1.0#X509PKIPathv1
```

or a `ValueType` attribute of

```
http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-
profile-1.0#PKCS7
```

[WS-SECX509] containing the requestor's long-term public signing key and associated certificates. At least one included certificate SHALL have a subject name containing the NEMO node ID of the requestor and the transmitted subject public key. For transmission of certificate chains using PKIPath, the last certificate in the chain SHALL NOT be signed by the certificate's subject public key pair. The `<wsse:BinarySecurityToken>` element MAY contain a `nemosec:Usage` attribute with a value containing the value of the URI attribute in the `<nemosec:ProtocolDeclaration>` with the following fragment concatenated at the end:

```
#request-signingKey
```

3.1.4.1.1.1.8 Requestor's Public Encryption Key Certificates

The `<wsse:Security>` element MAY contain a `<wsse:BinarySecurityToken>` with a `ValueType` attribute of

```
http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-
profile-1.0#X509PKIPathv1
```

or a `ValueType` attribute of

```
http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-
profile-1.0#PKCS7
```

[WS-SECX509] containing the requestor's long-term public encryption key and associated certificates. At least one included certificate SHALL have a subject name containing the NEMO node ID of the requestor and the transmitted subject public key. For transmission of certificate chains using PKIPath, the last certificate in the chain SHALL NOT be signed by the certificate's subject public key pair. The `<wsse:BinarySecurityToken>` element MAY contain a `nemosec:Usage` attribute with a value containing the value of the URI attribute in the `<nemosec:ProtocolDeclaration>` with the following fragment concatenated at the end:

```
#response-encryptionKey
```

3.1.4.1.1.1.9 Requestor's Encrypted Message Encryption Key

The `<wsse:Security>` element SHALL contain an `<enc:EncryptedKey>` element containing the requestor's encrypted one-time message encryption key, as specified in §3.1.3.4. The `<enc:EncryptedKey>` element SHALL be identified by a `<nemosec:Reference>` element with a `nemosec:TargetUsage` attribute containing the value of the URI attribute in the `<nemosec:ProtocolDeclaration>` with the following fragment concatenated at the end:

```
#request-encryptedMessageKey
```

3.1.4.1.1.1.10 Signature

The `<wsse:Security>` element SHALL contain a `<ds:Signature>` element that contains a public-key signature using the requestor's signing key pair. The signature SHALL sign at least the protocol declaration, request message's timestamp, requestor's nonce, responder identifier, and the unencrypted requestor's message encrypting key. The signature SHOULD also include the requestor's public response encryption key certificates (to prevent key substitution attacks) and the `<S11:Body>` element or portions of it as required by the responder's policy. The signature SHALL include any `<wsse:SecurityTokenReference>` element within the `<wsse:Security>` element that contains the `nemosec:Usage` attribute of an element identified above for signature inclusion; the signature SHALL also include the target of any such `<wsse:SecurityTokenReference>` element. The signature SHALL be applied before parts are encrypted. The signature itself SHOULD be encrypted to prevent certain attacks. The `<ds:Signature>` element SHALL be identified by a `<nemosec:Reference>` element with a `nemosec:TargetUsage` attribute containing the value of the URI attribute in the `<nemosec:ProtocolDeclaration>` with the following fragment concatenated at the end:

```
#request-signature
```

3.1.4.1.1.2 Processing Rules

The requestor SHOULD validate the binding between the responder's node ID and the responder's public encryption key before sending the message.

The responder MAY decide to reject the message, for example to prevent denial of service attacks. The responder MUST reject the document if it does not understand the protocol identifier or the profile identifier (if present), or if the `<nemosec:Step>` element is present and is not as expected. If the responder decides to process the message, the responder MUST verify that the time the request message was received is not later than the time indicated in the `<wsu:Expires>` element of the timestamp, if any. The responder MAY reject the message based on its own timestamp policy, as well. The responder SHOULD check the requestor's nonce to see if the same nonce has been used before. (Note that the responder need only check against nonces received within its window of timestamp acceptance.) The responder SHALL check that the NEMO node ID contained in the `<nemosec:ToNode>` element in the security header is a NEMO node identity of the responder. The responder SHALL check that the unencrypted requestor's message encryption key that appeared in the message signature is the same as is used to encrypt the message.

The responder SHALL verify the signature in the request message, the responder SHALL validate the binding between the requestor's node ID and the signature key, and the responder SHALL verify that the signature signs at least the parts specified in §3.1.4.1.1.1.10. If the security header contains a `<nemosec:FromNode>` element, the responder SHALL verify that the element contains the requestor's node ID.

If the request message does not pass these checks, the responder MUST NOT process the request, and MAY generate a fault.

3.1.4.1.2 Response Message

3.1.4.1.2.1 Security Header

The response SOAP message transmitted from the responder (service, or service provider) to the requestor (client, or service consumer) SHALL contain a `<wsse:Security>` header [WS-SEC], with no actor or role attribute, that contains the following elements: protocol identifier, responder's timestamp, responder's nonce, requestor's nonce, requestor's identifier, responder's encrypted message encryption key, responder's certificates, and a signature. The `<wsse:Security>` element MAY contain a profile identifier.

3.1.4.1.2.1.1 Protocol Identifier

The `<wsse:Security>` element SHALL include a `<nemosec:ProtocolDeclaration>` element with a URI attribute having the value specified in §3.1.4.1.1.1.1.

The `<nemosec:ProtocolDeclaration>` element SHOULD include a `<nemosec:Step>` element with a type attribute containing the following string:

```
response
```

3.1.4.1.2.1.2 Responder's Timestamp

The `<wsse:Security>` element SHALL contain a `<wsu:Timestamp>` element that indicates the time at which the responder sends the response message. The `<wsu:Timestamp>` element MAY contain a `nemosec:Usage` attribute with a value containing the value of the URI attribute in the `<nemosec:ProtocolDeclaration>` with the following fragment concatenated at the end:

```
#response-timestamp
```

3.1.4.1.2.1.3 Responder's Identifier

The `<wsse:Security>` element MAY contain a `<nemosec:FromNode>` element containing the NEMO node ID of the responder. The `<nemosec:FromNode>` element MAY contain a `nemosec:Usage` attribute with a value containing the value of the URI attribute in the `<nemosec:ProtocolDeclaration>` with the following fragment concatenated at the end:

```
#response-fromNode
```

3.1.4.1.2.1.4 Requestor's Identifier

The `<wsse:Security>` element SHALL contain a `<nemosec:ToNode>` element containing the NEMO node ID of the requestor. The `<nemosec:ToNode>` element MAY contain a `nemosec:Usage` attribute with a value containing the value of the URI attribute in the `<nemosec:ProtocolDeclaration>` with the following fragment concatenated at the end:

```
#response-toNode
```

3.1.4.1.2.1.5 Self-encrypted Message Key

The `<wsse:Security>` element SHALL contain an `<enc:EncryptedData>` element encrypted with the message encryption key, containing a `<wsse:BinarySecurityToken>` element containing the cleartext symmetric message key. The `<wsse:BinarySecurityToken>` element SHOULD contain a `ValueType` attribute as specified in §3.1.3.5.7. The `<wsse:BinarySecurityToken>` element MAY contain a `nemosec:Usage` attribute with a value containing the value of the URI attribute in the `<nemosec:ProtocolDeclaration>` with the following fragment concatenated at the end:

```
#response-messageKey
```

3.1.4.1.2.1.6 Requestor's Nonce

The `<wsse:Security>` element SHALL contain a `<wsse:Nonce>` with a `ValueType` attribute as specified in §3.1.3.2, whose contents are the same as the contents of the requestor's nonce element in the corresponding request message. The `<wsse:Nonce>` element MAY contain a `nemosec:Usage` attribute with a value containing the value of the URI attribute in the `<nemosec:ProtocolDeclaration>` with the following fragment concatenated at the end:

```
#response-returnedNonce
```

3.1.4.1.2.1.7 Responder's Nonce

If a later confirmation message is expected from the requestor, the `<wsse:Security>` element SHALL contain a `<wsse:Nonce>` with a `ValueType` attribute as specified in §3.1.3.2 containing a base64-encoded octet string of no more than 128 octets generated by the responder. The `<wsse:Nonce>` element MAY contain a `nemosec:Usage` attribute with a value containing the value of the URI attribute in the `<nemosec:ProtocolDeclaration>` with the following fragment concatenated at the end:

```
#response-nonce
```

3.1.4.1.2.1.8 Responder's Public Signing Key Certificates

The `<wsse:Security>` element MAY contain a `<wsse:BinarySecurityToken>` with a `ValueType` attribute of

```
http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-
profile-1.0#X509PKIPathv1
```

or a `ValueType` attribute of

```
http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-
profile-1.0#PKCS7
```

[WS-SECX509] containing the responder's long-term public signing key and associated certificates. At least one included certificate SHALL have a subject name containing the NEMO node ID of the requestor and the transmitted subject public key. For transmission of certificate chains using PKIPath, the last certificate in the chain SHALL NOT be signed by the certificate's subject public key pair. The `<wsse:BinarySecurityToken>` element MAY contain a `nemosec:Usage` attribute with a value containing the value of the URI attribute in the `<nemosec:ProtocolDeclaration>` with the following fragment concatenated at the end:

```
#response-signingKey
```

3.1.4.1.2.1.9 Responder's Encrypted Message Encryption Key

The `<wsse:Security>` element SHALL contain an `<enc:EncryptedKey>` element containing the responder's encrypted message encryption key, as specified in §3.1.3.4. The `<enc:EncryptedKey>` element SHALL be identified by a `<nemosec:Reference>` element with a `nemosec:TargetUsage` attribute containing the value of the URI attribute in the `<nemosec:ProtocolDeclaration>` with the following fragment concatenated at the end:

```
#response-encryptedMessageKey
```

3.1.4.1.2.1.10 Signature

The `<wsse:Security>` element SHALL contain a `<ds:Signature>` element that contains a public-key signature using the responder's signing key pair. The signature SHALL sign at least the protocol declaration, response message's timestamp, requestor's nonce, responder's nonce, requestor identifier, and the unencrypted responder's message encryption key. The signature SHOULD also include the `<S11:Body>` element or portions of it as required by the responder's policy. The signature SHALL include any `<wsse:SecurityTokenReference>` element within the `<wsse:Security>` element that contains the `nemosec:Usage` attribute of an element identified above for signature inclusion; the signature SHALL also include the target of any such `<wsse:SecurityTokenReference>` element. The signature SHALL be applied before parts are encrypted. The signature itself SHOULD be encrypted to prevent certain attacks. The `<ds:Signature>` element SHALL be identified by a `<nemosec:Reference>` element with a `nemosec:TargetUsage` attribute containing the value of the URI attribute in the `<nemosec:ProtocolDeclaration>` with the following fragment concatenated at the end:

```
#response-signature
```

3.1.4.1.2.2 Processing Rules

The responder SHOULD validate the binding between the requestor's node ID and the requestor's public encryption key before sending the message.

The requestor MAY decide to reject the message, for example to prevent denial of service attacks. The requestor MUST reject the document if it does not understand the protocol identifier or the profile identifier (if present), or if the `<nemosec:Step>` element is present and is not as expected. If the requestor decides to process the message, the requestor MUST verify that the time the response message was received is not later than the time indicated in the `<wsu:Expires>` element of the timestamp, if any. The requestor MAY reject the message based on its own timestamp policy, as well. The requestor SHALL check the requestor's nonce to see if it is the same value as was sent in the corresponding request message. The requestor MAY check the responder's nonce to see if the same nonce has been used before. (Note that the requestor need only check against nonces received within its window of timestamp acceptance.) The requestor SHALL check that the NEMO node ID contained in the `<nemosec:ToNode>` element in the security header is a NEMO node identity of the requestor. The requestor SHALL check that the unencrypted responder's message encryption key that appeared in the message signature is the same as is used to encrypt the message.

The requestor SHALL verify the signature in the response message, the requestor SHALL validate the binding between the responder's node ID and the signature key, and the requestor SHALL verify that the signature signs at least parts specified in §3.1.4.1.2.1.10. If the security header contains a `<nemosec:FromNode>` element, the requestor SHALL verify that the element contains the responder's node ID.

If the response message does not pass these checks, the requestor MUST NOT process the response.

3.1.4.1.3 Confirmation Message

3.1.4.1.3.1 Security Header

The confirmation SOAP message transmitted from the requestor (client, or service consumer) to the responder (service, or service provider) SHALL contain a `<wsse:Security>` header [WS-SEC], with no actor or role attribute, that contains the following elements: protocol identifier, requestor's timestamp, responder's nonce, responder's identifier, self-encrypted message key, requestor's encrypted one-time message encryption key, and a signature. The security header MAY also contain the requestor's public signing key certificates, SAML assertions [SAML1.1] and a profile identifier.

3.1.4.1.3.1.1 Protocol Identifier

The `<wsse:Security>` element SHALL include a `<nemosec:ProtocolDeclaration>` element with a URI having the value specified in §3.1.4.1.1.1.1.

The `<nemosec:ProtocolDeclaration>` element SHOULD include a `<nemosec:Step>` element with a `Type` attribute containing the following string:

```
confirmation
```

3.1.4.1.3.1.2 Requestor's Timestamp

The `<wsse:Security>` element SHALL contain a `<wsu:Timestamp>` element that indicates the time at which the requestor sends the confirmation message. The `<wsu:Timestamp>` element MAY contain a `nemosec:Usage` attribute with a value containing the value of the URI attribute in the `<nemosec:ProtocolDeclaration>` with the following fragment concatenated at the end:

```
#confirmation-timestamp
```

3.1.4.1.3.1.3 Responder's Nonce

The `<wsse:Security>` element SHALL contain a `<wsse:Nonce>`, as specified in §3.1.3.2, containing the responder's nonce passed in the corresponding response message. The `<wsse:Nonce>` element MAY contain a `nemosec:Usage` attribute with a value containing the value of the URI attribute in the `<nemosec:ProtocolDeclaration>` with the following fragment concatenated at the end:

```
#confirmation-returnedNonce
```

3.1.4.1.3.1.4 Requestor’s Identifier

The `<wsse:Security>` element MAY contain a `<nemosec:FromNode>` element containing the NEMO node ID of the requestor. The `<nemosec:FromNode>` element MAY contain a `nemosec:Usage` attribute with a value containing the value of the URI attribute in the `<nemosec:ProtocolDeclaration>` with the following fragment concatenated at the end:

```
#confirmation-fromNode
```

3.1.4.1.3.1.5 Responder’s Identifier

The `<wsse:Security>` element SHALL contain a `<nemosec:ToNode>` element containing the NEMO node ID of the responder. The `<nemosec:ToNode>` element MAY contain a `nemosec:Usage` attribute with a value containing the value of the URI attribute in the `<nemosec:ProtocolDeclaration>` with the following fragment concatenated at the end:

```
#confirmation-toNode
```

3.1.4.1.3.1.6 Self-encrypted Message Key

The `<wsse:Security>` element SHALL contain an <enc:EncryptedData> element, encrypted with the message encryption key, containing a `<wsse:BinarySecurityToken>` element containing the cleartext symmetric message key. The `<wsse:BinarySecurityToken>` element SHOULD contain a `ValueType` attribute as specified in §3.1.3.5.7. The `<wsse:BinarySecurityToken>` element MAY contain a `nemosec:Usage` attribute with a value containing the value of the URI attribute in the `<nemosec:ProtocolDeclaration>` with the following fragment concatenated at the end:

```
#confirmation-messageKey
```

3.1.4.1.3.1.7 Requestor’s Encrypted Message Encryption Key

The `<wsse:Security>` element SHALL contain an <enc:EncryptedKey> element containing the requestor’s encrypted one-time message encryption key, as specified in §3.1.3.4. The <enc:EncryptedKey> element SHALL be identified by a `<nemosec:Reference>` element with a `nemosec:TargetUsage` attribute containing the value of the URI attribute in the `<nemosec:ProtocolDeclaration>` with the following fragment concatenated at the end:

```
#confirmation-encryptedMessageKey
```

3.1.4.1.3.1.8 Requestor’s Public Signing Key Certificates

The `<wsse:Security>` element MAY contain a `<wsse:BinarySecurityToken>` with a `ValueType` attribute of

```
http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-
profile-1.0#X509PKIPathv1
```

or a `ValueType` attribute of

```
http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-
profile-1.0#PKCS7
```

[WS-SECX509] containing the requestor's long-term public signing key and associated certificates. At least one included certificate SHALL have a subject name containing the NEMO node ID of the requestor and the transmitted subject public key. For transmission of certificate chains using PKIPath, the last certificate in the chain SHALL NOT be signed by the certificate's subject public key pair. The `<wsse:BinarySecurityToken>` element MAY contain a `nemosec:Usage` attribute with a value containing the value of the URI attribute in the `<nemosec:ProtocolDeclaration>` with the following fragment concatenated at the end:

```
#confirmation-signingKey
```

3.1.4.1.3.1.9 Signature

The `<wsse:Security>` element SHALL contain a `<ds:Signature>` element that contains a public-key signature using the requestor's signing key pair. The signature SHALL sign at least the protocol declaration, request message's timestamp, responder's nonce, responder identifier, and the unencrypted requestor's message encrypting key. The signature SHOULD also include the `<S11:Body>` element or portions of it as required by the responder's policy. The signature SHALL include any `<wsse:SecurityTokenReference>` element within the `<wsse:Security>` element that contains the `nemosec:Usage` attribute of an element identified above for signature inclusion; the signature SHALL also include the target of any such `<wsse:SecurityTokenReference>` element. The signature SHALL be applied before parts are encrypted. The signature itself SHOULD be encrypted to prevent certain attacks. The `<ds:Signature>` element SHALL be identified by a `<nemosec:Reference>` element with a `nemosec:TargetUsage` attribute containing the value of the URI attribute in the `<nemosec:ProtocolDeclaration>` with the following fragment concatenated at the end:

```
#confirmation-signature
```

3.1.4.1.3.2 Processing Rules

The requestor SHOULD validate the binding between the responder's node ID and the responder's public encryption key before sending the message.

The responder MAY decide to reject the message, for example to prevent denial of service attacks. The responder MUST reject the document if it does not understand the protocol identifier or the profile identifier (if present), or if the `<nemosec:Step>` element is present and is not as expected. If the responder decides to process the message, the responder MUST verify that the time the request message was received is not later than the time indicated in the `<wsu:Expires>` element of the timestamp, if any. The responder MAY reject the message based on its own timestamp policy, as well. The responder SHOULD check the responder's nonce to see if it is the same as the nonce that was passed to the requestor in the corresponding response message. The responder SHALL check that the NEMO node ID contained in the `<nemosec:ToNode>` element in the security header is a NEMO node identity of the responder. The responder SHALL check that the unencrypted requestor's message encryption key that appeared in the message signature is the same as is used to encrypt the message.

The responder SHALL verify the signature in the confirmation message, the responder SHALL validate the binding between the requestor's node ID and the signature key, and the responder SHALL verify that the signature signs at least the parts specified in §3.1.4.1.3.1.9. If the security header contains a `<nemosec:FromNode>` element, the responder SHALL verify that the element contains the requestor's node ID. The responder SHALL verify that the requestor's node ID matches the requestor node ID in the correlated request message.

If the request message does not pass these checks, the responder MUST NOT process the confirmation.

3.1.4.1.4 Fault Response Message

The responder MAY respond to a request with an error message, indicating a premature termination of the protocol. Such a message SHALL conform to specifications for a fault response (§2), and MAY contain a `<nemosec:ProtocolDeclaration>` element containing a `<nemosec:Step>` element with a `Type` attribute containing the predefined type string

```
request-fault
```

Fault step types MUST NOT be used if the security protocol is not abnormally terminated, even if the message body contains an `<S11:Fault>` element. Service access authorization is not part of the NEMO Basic Secure Messaging Protocol, so authorization failures MUST NOT be reported using fault step types.

A message with a secure protocol step type that is a fault secure protocol step type MAY be protected with a signature and encryption, using similar syntax and semantics as are used for a touchdown response message. However, processors must assume that a fault step type indicates a failure of some condition of the secure protocol, so the normal security conditions may not apply. A NEMO node that sends a message with a fault secure protocol step type MAY choose not to encrypt or sign the fault message, despite service description policies that specify the use of cryptographic protections. It is RECOMMENDED that messages with a fault secure protocol step type not contain sensitive information if the authentication of the recipient or other relevant security properties are in doubt.

Within a fault message, the `nemosec:Usage` attribute of NEMO-specified security tokens and the `nemosec:TargetUsage` attribute of `ProtocolDeclaration/Reference` elements SHOULD be the same as those of the corresponding response message, with the string "`response`" occurring in the URI fragment replaced with the string "`fault`". For example, a `<nemosec:toNode>` element occurring in a fault response would be assigned the `nemosec:Usage` attribute of

```
#fault-toNode
```

3.1.4.1.5 Responder's Public Encryption Key

A security token containing the responder's public key used to encrypt a request message MAY contain a `nemosec:Usage` attribute with a value containing the value of the URI attribute in the `<nemosec:ProtocolDeclaration>` with the following fragment concatenated at the end:

```
#request-encryptionKey
```

3.1.4.2 Other Protection Levels

In the basic protocol, integrity, confidentiality, and freshness protections are optional. Requirements for these protections can be expressed in policy.

3.1.4.2.1 Freshness Protections

If timestamp and nonce elements are not to be used within a message, they SHOULD NOT appear in the message. The processing rules SHALL remain the same as for the full security case, except for the checking of nonces and/or timestamps.

Note: If nonces are used without timestamps, this specification does not specify how long receiver nodes should remember the nonces that have been used to date.

3.1.4.2.2 Integrity Protection

If integrity protection is not to be used within a message, the signature element and the sender's certificates SHOULD NOT appear in the `<wsse:Security>` element. The processing rules SHALL remain the same as for the full security case, except for the validating of the signature and the binding of the signing key to the sender's node ID. If integrity protection is not used, a message SHOULD contain the sender's NEMO node identifier in a `<nemosec:FromNode>` element.

3.1.4.2.3 Confidentiality Protection

If confidentiality protection is not to be used within a message, the message payload SHALL NOT be encrypted. The processing rules SHALL remain the same as for the full security case, except for the decrypting of the received message payload, and validating the binding between the receiver's node ID and public encryption key.

3.1.5 Recommended Fault Codes

The following fault codes are RECOMMENDED for use in the described situations.

Description	Recommended Fault Code
Unrecognized elements or extensions	wsse:UnsupportedSecurityToken
Requested secure session lifetime unacceptable	wsse:InvalidSecurity
Unrecognized protocol	nemosec:UnsupportedSecureProtocol
Unrecognized profile	nemosec:UnsupportedSecureProtocol
Unrecognized protocol step, sequence number, or message element	wsse:UnsupportedSecurityToken
Expired message (according to timestamp `<wsu:Expires>` element)	wsu:MessageExpired
Expired message (according to receiver's timestamp policy)	wsu:MessageExpired
Reuse of a nonce	wsse:InvalidSecurity
Returned nonce mismatch	wsse:InvalidSecurity
Incorrect ToNode	wsse:InvalidSecurityToken

Mismatch of message key, as provided in an EncryptedKey (used to decrypt the received message) and as provided in an EncryptedData	wsse:InvalidSecurity
Duplicate wsu:Id attributes	soap:Client
Signature validation failed	wsse:FailedCheck
Signing key authentication failed	wsse:FailedAuthentication
Protocol-required elements left out of signature	wsse:InvalidSecurity
Failure to authentication encryption key.	wsse:FailedAuthentication

3.1.6 Examples

3.1.6.1 Basic Secure Messaging Protocol

3.1.6.1.1 Request Message

```
<?xml version="1.0" encoding="UTF-8"?>
<S11:Envelope
   xmlns:S11="http://schemas.xmlsoap.org/soap/envelope/">
   <S11:Header>
      <wsse:Security
         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-
wss-wssecurity-utility-1.0.xsd"
         xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
         xmlns:enc="http://www.w3.org/2001/04/xmlenc#"
         xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#"
         xmlns:nemosec=
           "http://nemo.intertrust.com/2005/10/security"
         S11:mustUnderstand="1">

         <!-- Protocol declaration -->
         <nemosec:ProtocolDeclaration
            URI="http://nemo.intertrust.com/2005/10/security/secure-
protocol/basic/1.0"
            wsu:Id="NEMO_ID4"

   nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
protocol">
            <nemosec:Step Type="request"/>
            <nemosec:Reference

   nemosec:TargetUsage="http://nemo.intertrust.com/2005/10/security/secu
re-protocol/basic/1.0#request-encryptedMessageKey"
                  URI="#NEMO_ID1"/>
            <nemosec:Reference

   nemosec:TargetUsage="http://nemo.intertrust.com/2005/10/security/secu
re-protocol/basic/1.0#request-signature"
                  URI="#NEMO_ID3"/>
         </nemosec:ProtocolDeclaration>

         <!-- Timestamp -->
         <wsu:Timestamp
            wsu:Id="NEMO_ID6"

   nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
protocol/basic/1.0#request-timestamp">
            <wsu:Created>2005-06-15T02:08:21.372</wsu:Created>
            <wsu:Expires>2005-06-15T03:08:21.372</wsu:Expires>
         </wsu:Timestamp>

         <!-- Nonce -->
         <wsse:Nonce
            wsu:Id="NEMO_ID7"

   nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
protocol/basic/1.0#request-nonce"
            >UU4oOCriFXA=</wsse:Nonce>

         <!-- ToNode -->
         <nemosec:ToNode
            wsu:Id="NEMO_ID8"

   nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
protocol/basic/1.0#request-toNode"
            >urn:nemo:node:ObjectProvider</nemosec:ToNode>

         <!-- FromNode -->
         <nemosec:FromNode
            wsu:Id="NEMO_ID9"

   nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
protocol/basic/1.0#request-fromNode"

   >1.3.6.1.4.1.7584.1.1.1=urn:nemo:node:Device</nemosec:FromNode>

         <!-- Encrypted message key -->
         <enc:EncryptedKey
            Id="NEMO_ID1">
            <enc:EncryptionMethod
             Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-
mgf1p"/>
            <enc:CipherData>
             <enc:CipherValue>wboB...Ii4=</enc:CipherValue>
            </enc:CipherData>
            <enc:ReferenceList>
             <!-- Encrypted signature element -->
```

```
              <enc:DataReference URI="#NEMO_ID15"/>
              <!-- Encrypted body contents -->
              <enc:DataReference URI="#NEMO_ID16"/>
            </enc:ReferenceList>
         </enc:EncryptedKey>

         <!-- Self-encrypted message key -->
         <wsse:BinarySecurityToken

   ValueType="http://nemo.intertrust.com/2005/10/security/BST/SymmetricK
ey"
   EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-soap-message-security-1.0#Base64Binary"
            wsu:Id="NEMO_ID10"

   nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
protocol/basic/1.0#request-messageKey"
            >/1KLw02aJ73ByZ1ya1ZFvw==</wsse:BinarySecurityToken>

         <!-- Message signing certificate chain -->
         <wsse:BinarySecurityToken
            ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-x509-token-profile-1.0#X509PKIPathv1"
   EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-soap-message-security-1.0#Base64Binary"
            wsu:Id="NEMO_ID2"

   nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
protocol/basic/1.0#request-signingKey"
            >MIIC...xUM=</wsse:BinarySecurityToken>

         <!-- Signature -->
         <ds:Signature
            Id="NEMO_ID3">
            <ds:SignedInfo>
              <ds:CanonicalizationMethod
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                  <ec:InclusiveNamespaces/>
              </ds:CanonicalizationMethod>
              <ds:SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>

              <!-- Protocol declaration -->
              <ds:Reference URI="#NEMO_ID4">
                  <ds:Transforms>
                      <ds:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                        <ec:InclusiveNamespaces/>
                      </ds:Transform>
                  </ds:Transforms>
                  <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
```

```

   <ds:DigestValue>nI8PbZSH68Hs85fnEhFu4tmbutQ=</ds:DigestValue>
              </ds:Reference>

   <ds:DigestValue>c84XiUsfpEk9YUHeMFtyd7l/wsw=</ds:DigestValue>
              </ds:Reference>

              <!-- Timestamp -->
              <ds:Reference URI="#NEMO_ID6">
                  <ds:Transforms>
                      <ds:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                        <ec:InclusiveNamespaces/>
                      </ds:Transform>
                  </ds:Transforms>
                  <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

   <ds:DigestValue>X/b8LS+q9YaLIiJbidJLLyzNsR4=</ds:DigestValue>
              </ds:Reference>

              <!-- Nonce -->
              <ds:Reference URI="#NEMO_ID7">
                  <ds:Transforms>
                      <ds:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                        <ec:InclusiveNamespaces/>
                      </ds:Transform>
                  </ds:Transforms>
                  <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

   <ds:DigestValue>96l29nkFs+ZYc8JHESTZRDXLlwo=</ds:DigestValue>
              </ds:Reference>

              <!-- ToNode -->
              <ds:Reference URI="#NEMO_ID8">
                  <ds:Transforms>
                      <ds:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                        <ec:InclusiveNamespaces/>
                      </ds:Transform>
                  </ds:Transforms>
                  <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

   <ds:DigestValue>FtoLVhdlmc3vY/v9DOEnj94O8/I=</ds:DigestValue>
              </ds:Reference>

              <!-- FromNode -->
              <ds:Reference URI="#NEMO_ID9">
                  <ds:Transforms>
```

```
                      <ds:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                        <ec:InclusiveNamespaces/>
                      </ds:Transform>
                  </ds:Transforms>
                  <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

   <ds:DigestValue>Xh5nY1FqNoGA9RqJuBJN19Lg6G8=</ds:DigestValue>
              </ds:Reference>

              <!-- Self-encrypted message key -->
              <ds:Reference URI="#NEMO_ID10">
                  <ds:Transforms>
                      <ds:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                        <ec:InclusiveNamespaces/>
                      </ds:Transform>
                  </ds:Transforms>
                  <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

   <ds:DigestValue>LvALWu5ULqUaHRefe0NWQTNShBM=</ds:DigestValue>
              </ds:Reference>

              <!-- Response encryption key -->
              <ds:Reference URI="#NEMO_ID13">
                  <ds:Transforms>
                      <ds:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                        <ec:InclusiveNamespaces/>
                      </ds:Transform>
                  </ds:Transforms>
                  <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

   <ds:DigestValue>4jkyhQYEu1gsrocogUCHPsl1NwE=</ds:DigestValue>
              </ds:Reference>

              <!-- SOAP Body -->
              <ds:Reference URI="#NEMO_ID14">
                  <ds:Transforms>
                      <ds:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                        <ec:InclusiveNamespaces/>
                      </ds:Transform>
                  </ds:Transforms>
                  <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

   <ds:DigestValue>oQM3cTcTn2WYP0F6ac6+dwWJnfA=</ds:DigestValue>
              </ds:Reference>
```

```
            </ds:SignedInfo>
            <ds:SignatureValue>wBbi...jvA=</ds:SignatureValue>
            <ds:KeyInfo>
              <wsse:SecurityTokenReference>
                  <!-- Message signing certificate chain -->
                  <wsse:Reference URI="#NEMO_ID2"/>
              </wsse:SecurityTokenReference>
            </ds:KeyInfo>
         </ds:Signature>

         <!-- Response encryption key certificate chain -->
         <wsse:BinarySecurityToken
            ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-x509-token-profile-1.0#X509PKIPathv1"
   EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-soap-message-security-1.0#Base64Binary"
            wsu:Id="NEMO_ID13"

   nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
protocol/basic/1.0#response-encryptionKey"
            >MIIC...Wgc=</wsse:BinarySecurityToken>

         <!-- Role assertion -->
         <saml:Assertion
xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
AssertionID="ID1117583305016" IssueInstant="2005-05-31T23:48:24Z"
Issuer="1.3.6.1.4.1.7584.1.1.1=urn:nemo:node:ObjectAuthority"
MajorVersion="1" MinorVersion="1">
            <saml:Conditions NotBefore="2005-05-31T23:48:24Z"
NotOnOrAfter="2006-05-31T23:48:24Z"/>
            <saml:AttributeStatement>
              <saml:Subject>
                  <saml:NameIdentifier
Format="urn:oasis:names:tc:SAML:1.1:nameid-
format:X509SubjectName">1.3.6.1.4.1.7584.1.1.1=urn:nemo:node:Device</sam
l:NameIdentifier>
              </saml:Subject>
              <saml:Attribute AttributeName="role"
AttributeNamespace="http://nemo.intertrust.com/2004/attribute">

   <saml:AttributeValue>ObjectProviderClient</saml:AttributeValue>
              </saml:Attribute>
            </saml:AttributeStatement>
            <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#">
                       <ec:InclusiveNamespaces/>
                  </ds:CanonicalizationMethod>

                  <ds:SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
```

```
                  <ds:Reference URI="#ID1117583305016">
                      <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#">
                        <ec:InclusiveNamespaces/>
                      </ds:Transform>
                      </ds:Transforms>
                      <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

   <ds:DigestValue>0oKaxQ/MPAZ/bjtNT2I/EiIfVi8=</ds:DigestValue>
                  </ds:Reference>
              </ds:SignedInfo>
              <ds:SignatureValue>Fb9C...ytM=</ds:SignatureValue>
              <ds:KeyInfo>
                  <ds:X509Data>

   <ds:X509Certificate>MIIC...cwE=</ds:X509Certificate>
                  </ds:X509Data>
              </ds:KeyInfo>
            </ds:Signature>
         </saml:Assertion>
      </wsse:Security>
   </S11:Header>

   <!-- SOAP Body -->
   <S11:Body
      xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-utility-1.0.xsd"
      wsu:Id="NEMO_ID14">

      <!-- Payload -->
      <ObjectRequestPayload xmlns="urn:services:provide-objects:schema">
      </ObjectRequestPayload>
   </S11:Body>
</S11:Envelope>
```

3.1.6.1.2 Response Message

```
<?xml version="1.0" encoding="UTF-8"?>
<S11:Envelope
   xmlns:S11="http://schemas.xmlsoap.org/soap/envelope/">
   <S11:Header>
      <wsse:Security
         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-
wss-wssecurity-utility-1.0.xsd"
         xmlns:enc="http://www.w3.org/2001/04/xmlenc#"
         xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
         xmlns:nemosec=
           "http://nemo.intertrust.com/2005/10/security"
         xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#"
         S11:mustUnderstand="1">

         <!-- Protocol declaration -->
         <nemosec:ProtocolDeclaration
            URI="http://nemo.intertrust.com/2005/10/security/secure-
protocol/basic/1.0"
            wsu:Id="NEMO_ID4"

   nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
protocol">
            <nemosec:Step Type="response"/>
            <nemosec:Reference

   nemosec:TargetUsage="http://nemo.intertrust.com/2005/10/security/secu
re-protocol/basic/1.0#response-encryptedMessageKey"
                  URI="#NEMO_ID1"/>
            <nemosec:Reference

   nemosec:TargetUsage="http://nemo.intertrust.com/2005/10/security/secu
re-protocol/basic/1.0#response-signature"
                  URI="#NEMO_ID3"/>
         </nemosec:ProtocolDeclaration>

         <!-- Timestamp -->
         <wsu:Timestamp
            wsu:Id="NEMO_ID6"

   nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
protocol/basic/1.0#response-timestamp">
            <wsu:Created>2005-06-15T02:08:21.95</wsu:Created>
            <wsu:Expires>2005-06-15T03:08:21.95</wsu:Expires>
         </wsu:Timestamp>

         <!-- Nonce -->
         <wsse:Nonce
            wsu:Id="NEMO_ID7"

   nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
protocol/basic/1.0#response-nonce"
            >gFyj9hobPcY=</wsse:Nonce>

         <!-- Returned nonce -->
         <wsse:Nonce
            wsu:Id="NEMO_ID8"

   nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
protocol/basic/1.0#response-returnedNonce"
            >UU4oOCriFXA=</wsse:Nonce>

         <!-- ToNode -->
         <nemosec:ToNode
            wsu:Id="NEMO_ID9"

   nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
protocol/basic/1.0#response-toNode"
            >urn:nemo:node:Device</nemosec:ToNode>

         <!-- FromNode -->
         <nemosec:FromNode
            wsu:Id="NEMO_ID10"

   nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
protocol/basic/1.0#response-fromNode"

   >1.3.6.1.4.1.7584.1.1.1=urn:nemo:node:ObjectProvider</nemosec:FromNod
e>

         <!-- Encrypted message key -->
         <enc:EncryptedKey
            Id="NEMO_ID1">
            <enc:EncryptionMethod
             Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-
mgf1p"/>
            <enc:CipherData>
             <enc:CipherValue>qXLi...Pr4=</enc:CipherValue>
            </enc:CipherData>
            <enc:ReferenceList>
             <enc:DataReference URI="#NEMO_ID15"/>
             <enc:DataReference URI="#NEMO_ID16"/>
            </enc:ReferenceList>
         </enc:EncryptedKey>

         <!-- Self-encrypted message key -->
         <wsse:BinarySecurityToken

   ValueType="http://nemo.intertrust.com/2005/10/security/BST/SymmetricK
ey"
   EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-soap-message-security-1.0#Base64Binary"
            wsu:Id="NEMO_ID11"

   nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
protocol/basic/1.0#response-messageKey"
            >r40Md5PL7psz5V4KLNg8oQ==</wsse:BinarySecurityToken>

         <!-- Message signing certificate chain -->
         <wsse:BinarySecurityToken
            ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-x509-token-profile-1.0#X509PKIPathv1"
   EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-soap-message-security-1.0#Base64Binary"
```

```
            wsu:Id="NEMO_ID2"

   nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
protocol/basic/1.0#response-signingKey"
            >MIIC...xUM=</wsse:BinarySecurityToken>

         <!-- Signature -->
         <ds:Signature
            Id="NEMO_ID3">
            <ds:SignedInfo>
              <ds:CanonicalizationMethod
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                  <ec:InclusiveNamespaces/>
              </ds:CanonicalizationMethod>

              <ds:SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>

              <!-- Protocol declaration -->
              <ds:Reference URI="#NEMO_ID4">
                  <ds:Transforms>
                      <ds:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                        <ec:InclusiveNamespaces/>
                      </ds:Transform>
                  </ds:Transforms>
                  <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

   <ds:DigestValue>47xQ5MHQOobKhmiTz26dSE0IlXw=</ds:DigestValue>
              </ds:Reference>

   <ds:DigestValue>c84XiUsfpEk9YUHeMFtyd7l/wsw=</ds:DigestValue>
              </ds:Reference>

              <!-- Timestamp -->
              <ds:Reference URI="#NEMO_ID6">
                  <ds:Transforms>
                      <ds:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                        <ec:InclusiveNamespaces/>
                      </ds:Transform>
                  </ds:Transforms>
                  <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

   <ds:DigestValue>RLx5nlpdPLdoVckxj9TgH/7mZY0=</ds:DigestValue>
              </ds:Reference>

              <!-- Nonce -->
              <ds:Reference URI="#NEMO_ID7">
                  <ds:Transforms>
```

```
                      <ds:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                        <ec:InclusiveNamespaces/>
                      </ds:Transform>
                  </ds:Transforms>
                  <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

   <ds:DigestValue>I3IeanHd42nMhBt7QdDNWvPOE0w=</ds:DigestValue>
              </ds:Reference>

              <!-- Returned nonce -->
              <ds:Reference URI="#NEMO_ID8">
                  <ds:Transforms>
                      <ds:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                        <ec:InclusiveNamespaces/>
                      </ds:Transform>
                  </ds:Transforms>
                  <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

   <ds:DigestValue>EJU0huKed7KSAQWnz4MdXZQHZmI=</ds:DigestValue>
              </ds:Reference>

              <!-- ToNode -->
              <ds:Reference URI="#NEMO_ID9">
                  <ds:Transforms>
                      <ds:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                        <ec:InclusiveNamespaces/>
                      </ds:Transform>
                  </ds:Transforms>
                  <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

   <ds:DigestValue>mLKsYekEKNRiydR/JW//nyr1Jhc=</ds:DigestValue>
              </ds:Reference>

              <!-- FromNode -->
              <ds:Reference URI="#NEMO_ID10">
                  <ds:Transforms>
                      <ds:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                        <ec:InclusiveNamespaces/>
                      </ds:Transform>
                  </ds:Transforms>
                  <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

   <ds:DigestValue>3W7L7XXuqEr9azm7wpgksZ3TZjI=</ds:DigestValue>
              </ds:Reference>

              <!-- Message key -->
              <ds:Reference URI="#NEMO_ID11">
                  <ds:Transforms>
                      <ds:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                        <ec:InclusiveNamespaces/>
                      </ds:Transform>
                  </ds:Transforms>
                  <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

   <ds:DigestValue>jaLOXfkP07TOaU+LoOD/m3x7DNo=</ds:DigestValue>
              </ds:Reference>

              <!-- SOAP Body -->
              <ds:Reference URI="#NEMO_ID14">
                  <ds:Transforms>
                      <ds:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                        <ec:InclusiveNamespaces/>
                      </ds:Transform>
                  </ds:Transforms>
                  <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

   <ds:DigestValue>L/wM/Au+zdkRCyOO7IIBLJC84qM=</ds:DigestValue>
              </ds:Reference>
            </ds:SignedInfo>
            <ds:SignatureValue>U/rj...R1Q=</ds:SignatureValue>
            <ds:KeyInfo>
              <wsse:SecurityTokenReference>
                  <!-- Message signing certificate chain -->
                  <wsse:Reference URI="#NEMO_ID2"/>
              </wsse:SecurityTokenReference>
            </ds:KeyInfo>
         </ds:Signature>
      </wsse:Security>
   </S11:Header>

   <!-- SOAP Body -->
   <S11:Body
      xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-utility-1.0.xsd"
      wsu:Id="NEMO_ID14">

      <!-- Message payload -->
      <ObjectResponsePayload
         xmlns="urn:services:provide-objects:schema">
      </ObjectResponsePayload>
   </S11:Body>
</S11:Envelope>
```

3.1.6.1.3 Confirmation Message

```
<?xml version="1.0" encoding="UTF-8"?>
<S11:Envelope
   xmlns:S11="http://schemas.xmlsoap.org/soap/envelope/">
   <S11:Header>
      <wsse:Security
         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-
wss-wssecurity-utility-1.0.xsd"
         xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
         xmlns:enc="http://www.w3.org/2001/04/xmlenc#"
         xmlns:nemosec=
           "http://nemo.intertrust.com/2005/10/security"
         xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#"
         S11:mustUnderstand="1">

         <!-- Protocol declaration -->
         <nemosec:ProtocolDeclaration
            URI="http://nemo.intertrust.com/2005/10/security/secure-
protocol/basic/1.0"

   nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
protocol">
            <nemosec:Step Type="confirmation"/>
            <nemosec:Reference

   nemosec:TargetUsage="http://nemo.intertrust.com/2005/10/security/secu
re-protocol/basic/1.0#confirmation-encryptedMessageKey"
                  URI="#NEMO_ID1"/>
            <nemosec:Reference

   nemosec:TargetUsage="http://nemo.intertrust.com/2005/10/security/secu
re-protocol/basic/1.0#confirmation-signature"
                  URI="#NEMO_ID3"/>
         </nemosec:ProtocolDeclaration>

         <!-- Timestamp -->
         <wsu:Timestamp
            wsu:Id="NEMO_ID4"

   nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
protocol/basic/1.0#confirmation-timestamp">
            <wsu:Created>2005-06-15T02:08:22.2</wsu:Created>
            <wsu:Expires>2005-06-15T03:08:22.2</wsu:Expires>
         </wsu:Timestamp>

         <!-- Returned nonce -->
         <wsse:Nonce
            wsu:Id="NEMO_ID5"

   nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
protocol/basic/1.0#confirmation-returnedNonce"
            >gFyj9hobPcY=</wsse:Nonce>

         <!-- ToNode -->
         <nemosec:ToNode
            wsu:Id="NEMO_ID6"

   nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
protocol/basic/1.0#confirmation-toNode"
            >urn:nemo:node:ObjectProvider</nemosec:ToNode>

         <!-- FromNode -->
         <nemosec:FromNode
            wsu:Id="NEMO_ID7"

   nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
protocol/basic/1.0#confirmation-fromNode"

   >1.3.6.1.4.1.7584.1.1.1=urn:nemo:node:Device</nemosec:FromNode>

         <!-- Encrypted message key -->
         <enc:EncryptedKey
            Id="NEMO_ID1">
            <enc:EncryptionMethod
              Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-
mgf1p"/>
            <enc:CipherData>
              <enc:CipherValue>DJ38...4yM=</enc:CipherValue>
            </enc:CipherData>
            <enc:ReferenceList>
              <enc:DataReference URI="#NEMO_ID10"/>
              <enc:DataReference URI="#NEMO_ID11"/>
            </enc:ReferenceList>
         </enc:EncryptedKey>

         <!-- Self-encrypted message key -->
         <wsse:BinarySecurityToken

   ValueType="http://nemo.intertrust.com/2005/10/security/BST/SymmetricK
ey"
   EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-soap-message-security-1.0#Base64Binary"
            wsu:Id="NEMO_ID8"

   nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
protocol/basic/1.0#confirmation-messageKey"
            >/wuLG7KSi3HoHGak3Ibw7Q==</wsse:BinarySecurityToken>

         <!-- Message signing key certificate chain -->
         <wsse:BinarySecurityToken
```

```
            ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-x509-token-profile-1.0#X509PKIPathv1"
   EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-soap-message-security-1.0#Base64Binary"
            wsu:Id="NEMO_ID2"

   nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
protocol/basic/1.0#confirmation-signingKey"
            >MIIC....xUM=</wsse:BinarySecurityToken>

         <!-- Signature -->
         <ds:Signature
            Id="NEMO_ID3">
            <ds:SignedInfo>
              <ds:CanonicalizationMethod
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                  <ec:InclusiveNamespaces/>
              </ds:CanonicalizationMethod>

              <ds:SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>

              <!-- Timestamp -->
              <ds:Reference URI="#NEMO_ID4">
                  <ds:Transforms>
                      <ds:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                        <ec:InclusiveNamespaces/>
                      </ds:Transform>
                  </ds:Transforms>
                  <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

   <ds:DigestValue>CsZ4+1yUFTaZaN95saqjrYv4RRc=</ds:DigestValue>
              </ds:Reference>

              <!-- Nonce -->
              <ds:Reference URI="#NEMO_ID5">
                  <ds:Transforms>
                      <ds:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                        <ec:InclusiveNamespaces/>
                      </ds:Transform>
                  </ds:Transforms>
                  <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

   <ds:DigestValue>qo/2ozEhHKdXezzK41jlqpwFF/4=</ds:DigestValue>
              </ds:Reference>

              <!-- ToNode -->
              <ds:Reference URI="#NEMO_ID6">
```

```
                  <ds:Transforms>
                       <ds:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                         <ec:InclusiveNamespaces/>
                       </ds:Transform>
                  </ds:Transforms>
                  <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

   <ds:DigestValue>6WL57mr3KLwGZW6KhmC62sNfz9I=</ds:DigestValue>
              </ds:Reference>

              <!-- FromNode -->
              <ds:Reference URI="#NEMO_ID7">
                  <ds:Transforms>
                       <ds:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                         <ec:InclusiveNamespaces/>
                       </ds:Transform>
                  </ds:Transforms>
                  <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

   <ds:DigestValue>AWr11XUMv7jzEoy4JAuQwWQ6X0g=</ds:DigestValue>
              </ds:Reference>

              <!-- Self-encrypted message key -->
              <ds:Reference URI="#NEMO_ID8">
                  <ds:Transforms>
                       <ds:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                         <ec:InclusiveNamespaces/>
                       </ds:Transform>
                  </ds:Transforms>
                  <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

   <ds:DigestValue>ov+JClUnR6LBChe59TfDoQ0xJT4=</ds:DigestValue>
              </ds:Reference>

              <!-- SOAP Body -->
              <ds:Reference URI="#NEMO_ID9">
                  <ds:Transforms>
                       <ds:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                         <ec:InclusiveNamespaces/>
                       </ds:Transform>
                  </ds:Transforms>
                  <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

   <ds:DigestValue>TFg5gdgBYzBBKlk3jQBD5l6jxBw=</ds:DigestValue>
```

```
              </ds:Reference>
            </ds:SignedInfo>
            <ds:SignatureValue>6umm...ijk=</ds:SignatureValue>
            <ds:KeyInfo>
              <wsse:SecurityTokenReference>
                  <!-- Message signing certificate chain -->
                  <wsse:Reference URI="#NEMO_ID2"/>
              </wsse:SecurityTokenReference>
            </ds:KeyInfo>
         </ds:Signature>
      </wsse:Security>
   </S11:Header>

   <!-- SOAP Body -->
   <S11:Body
      wsu:Id="NEMO_ID9">
      <ObjectConfirmPayload xmlns="urn:services:provide-
objects:schema"/>
   </S11:Body>
</S11:Envelope>
```

3.2 *Security Considerations*

Message senders should be aware of the remote chance of a vulnerability to an element substitution attack. WS-Security [WS-SEC] recommends the use of bare-name (shortcut) XPointers to reference XML document elements. Using this mechanism, there is no way to tell from a signature alone if a referenced element was originally located within a message header, within the Security header, within the message body, or in some other location. However, only unusual message circumstances will create a vulnerability to such an attack.

4 NEMO Trust Management Bindings

4.1 *Introduction (Informative)*

This section describes bindings of NEMO trust management mechanisms—in particular, the use of SAML-specified URIs for NEMO node identifiers, the use of X.509 certificates for NEMO node authentication, the use of SAML attribute assertions with NEMO nodes, and the definition of a special NEMO node “role” attribute.

It is intended that NEMO roles be used in authorization decisions, both by service consumers and by service providers. If a role is asserted by an unknown authority, it may be necessary to authorize the role authority on the basis of the role authority’s own role. This implies a recursive chaining algorithm for verifying roles, based on role assertion authorization policy. Unlike other uses of authorization roles, NEMO roles are not organized in a privilege hierarchy, nor can they be quickly assumed and shed.

Also described is a means of simultaneously authenticating and authorizing a NEMO node via TLS server authentication. This binding is intended for use by small portable devices. This mechanism allows one-sided authentication and authorization using a TLS facility that is probably already present on the small device. However, use of this mechanism requires that certificate authorities issue certificates with authorization semantics. Also, a TLS client cannot signal to a TLS server the client’s preferred certificate trust anchor, so servers must offer only a single certificate chain, and clients must be provisioned to trust a sufficient set of certificate authorities.

4.2 *URI NEMO Identifier Binding*

NEMO nodes SHALL be identified by a canonicalized URI reference of length no more than 1024 characters. URI references are specified in [RFC2396]. The URI MUST be an absolute identifier (as opposed to a URI relative to a base identifier). The scheme of the URI MUST support a canonicalization algorithm, so that there is a unique canonicalized URI character sequence defined for the node identifier.

4.2.1 Canonical Representation of HTTP URLs

The canonical URI character sequence corresponding to an HTTP URL [RFC2616] SHALL observe the following properties.

- The URL SHALL NOT contain the default port for that URI-reference;
- Host names MUST be in lower case only;
- Scheme names MUST be in lower case only;
- The abs_path component MUST NOT be empty;
- Characters in the “reserved” and “unsafe” sets (see [RFC2396]) SHALL be given their `""%" HEX HEX"` encoding;
- Characters other than those in the “reserved” and “unsafe” sets (see [RFC2396]) SHALL NOT be given their `""%" HEX HEX"` encoding;

- Any `""%" HEX HEX"` encodings SHALL use lower case representations of the hexadecimal digits.

4.2.2 Canonical Representation of URNs

The canonical URI character sequence corresponding to a URN [RFC2141] SHALL observe the following properties.

- The leading "urn:" token SHALL be in lower case;
- The Namespace ID SHALL be in lower case;
- Characters in the "reserved" and "unsafe" sets (see [RFC2396]) SHALL be given their `""%" HEX HEX"` encoding;
- Characters other than those in the "reserved" and "unsafe" sets (see [RFC2396]) SHALL NOT be given their `""%" HEX HEX"` encoding;
- Any `""%" HEX HEX"` encodings SHALL use lowercase representations of the hexadecimal digits.
- The Namespace Specific String must be canonicalized according to any conventions defined by the Namespace ID.

4.3 *X.509 Authentication Binding*

4.3.1 X.500 Object Identifiers

Intertrust's private enterprise object identifier arc is given here.

```
id-itru OBJECT IDENTIFIER ::= {iso(1) identified-organization(3)
dod(6) internet(1) private(4) enterprise(1) intertrust(7584)}
```

This arc can be referenced as http://oid.elibel.tm.fr/1.3.6.1.4.1.7584 and is repeated here for your convenience.

4.3.1.1 NEMO

A sub-arc of the Intertrust object identifier arc is devoted to NEMO.

```
id-nemo OBJECT IDENTIFIER ::= {id-itru nemo(1)}
```

4.3.1.2 Name Attributes

The following arc is used within NEMO to identify X.500 name attributes, for use within X.500 distinguished names.

```
id-nemo-nat OBJECT IDENTIFIER ::= {id-nemo nameAttribute(1)}
```

4.3.1.3 Extended Key Usages

The following arc is used within NEMO to identify X.509 extended key usages within X.509 certificates.

```
id-nemo-xku OBJECT IDENTIFIER ::= {id-nemo extendedKeyUsage(2)}
```

4.3.2 X.509 Certificate Use

The public keys held by NEMO nodes SHALL be bound to the node identifiers via the use of a public key infrastructure, using X.509 certificate validation, as specified in [PKIX]. If the subject of an X.509 certificate is a NEMO node, the `subject` field SHALL contain a distinguished name, as specified in §4.3.4, and the `subjectAltName` extension MUST NOT be marked critical.

The subject key identifier extension of an X.509 certificate whose subject name is a NEMO node ID SHALL be a 160-bit SHA-1 hash of the `subjectPublicKey` field, this being the first method suggested by [RFC3280] section 4.2.1.2.

The X.509 extended key usage extension MAY be present and may be marked critical. NEMO nodes are not required to process the X.509 fields `issuerUniqueID` and `subjectUniqueID`.

4.3.3 Signature Algorithms

NEMO nodes MUST support the following certificate signature algorithms [PKIXALGS].

- sha1WithRSAEncryption

4.3.4 Distinguished Names

NEMO node distinguished names MUST contain exactly one distinguished name attribute, which SHALL be a “uri” attribute encoded as a UTF8String. The UTF-8 character sequence of the description distinguished name attribute MUST be an encoding of the canonical URI character sequence of the NEMO node identifier URI.

The uri attribute is defined using ASN.1 syntax and conventions in [X.520] as

```
uri ATTRIBUTE ::= {
   WITH SYNTAX UTF8String {ub-uri}
   EQUALITY MATCHING RULE caseExactMatch
   SUBSTRINGS MATCHING RULE caseExactSubstringsMatch
   ID id-nat-uri }

id-nat-uri OBJECT IDENTIFIER ::= {id-nemo-nat 1}
ub-uri INTEGER ::= 1024
```

4.3.5 Certificate Revocation

A NEMO node SHOULD NOT rely on or use any certificate identified in a valid certificate revocation list (CRL), as specified in [RFC3280].

4.3.6 Certificate and Key Renewal

NEMO nodes capable of secure time MUST NOT use or accept expired certificates for the purpose of authenticating NEMO nodes. A NEMO node MUST NOT use or accept for the purpose of authenticating a NEMO node a certificate whose expiration date is after the issue date of a valid CRL received from the certificate authority that issued the certificate.

A certificate authority (CA) MUST establish a rollover period before the CA’s certificate signing key expires. During the rollover period the CA MUST issue certificates signed with the new certificate signing key, as well as a pair of rollover certificates certifying the expiring key with the new key, and certifying the new key with the expiring key. The rollover certificates MUST expire at the same time as the CA’s older certificate signing key.

4.4 *SAML NEMO Node Attribute Assertions*

A NEMO node may be associated with trusted attributes. Attribute associations and values may be asserted by the issuance of `Assertion` elements containing `<saml:AttributeStatement>` elements, as specified in [SAML1.1]. SAML assertions MUST be signed. SAML assertions SHALL NOT inherit signatures from non-SAML elements ([SAML1.1], Section 5.3). If the `<saml:Subject>` element of a NEMO attribute assertion contains a `<saml:NameIdentifier>` element, the `Format` attribute of the `<saml:NameIdentifier>` element MUST be present, and MUST have the following value.

```
&nemo;/saml/name-format/uri
```

This format identifier indicates that the subject name identifier is a URI (see [RFC2396]). Also, the `Issuer` attribute of the `<saml:Assertion>` element MUST contain a URI. If the subject or issuer of a SAML assertion is a NEMO node, the content of the `<saml:NameIdentifier>` element or `Issuer` attribute SHALL be the URI identifier of the NEMO node.

Note (Informative): NEMO attributes are made trustworthy by the presence of trusted assertions. NEMO attributes may change with time, but NEMO attributes should not be variables that change faster than trusted assertions can be distributed.

4.4.1 Processing Rules for SAML Attributes

4.4.1.1 Attribute Association

The processing model of SAML attribute statements is not completely specified in [SAML1.1]. In the present NEMO binding, a SAML attribute statement signifies that the subject is associated with the attribute and with at least the indicated attribute values. Note the following:

1. In the present binding, two SAML attribute statements appearing in the same SAML assertion with the same subject and attribute designator are functionally equivalent to a single attribute statement with the combined set of attribute values.

2. Some policy processing rules may employ “negation as failure,” meaning that a policy succeeds if an attribute is not “known” by the processor (through the presence of attribute assertions in a message or in a processor cache) to be associated with a specific value. Under such processing rules, an attribute assertion that associates an attribute with two values is not functionally equivalent to two attribute assertions that separately associate the attribute with the same two values, since in the latter case a single assertion may be presented alone to the assertion processor, and the other withheld. For example, a policy might require an attribute to be associated with exactly one of the values x or y. The policy will succeed if only one of the assertions (x or y) is presented, but the policy will fail if only a combined assertion (x and y) is available.

3. In the present binding, there is no means, using only SAML attribute statements and without further specification, to assert that a subject is not associated with a given attribute value.

4. The specification of an attribute type may include further processing rules, such as restrictions on the number of associated values.

4.4.1.2 SAML Conditions

NEMO processors MUST process `NotBefore` and `NotOnOrAfter` attributes in a `<saml:Conditions>` element. Support for other SAML conditions is OPTIONAL. Processors MUST reject an assertion that contains unsupported conditions.

4.4.1.3 Validity Caching

Before relying on a SAML assertion, a NEMO processor SHOULD authenticate the issuer of the assertion. A NEMO processor MAY apply additional conditions and procedures before validating the reliability of a SAML assertion. For example, a processor may require that the issuer have a certain fixed identity, or else that the issuer be associated with certain attributes or attribute values.

Once a NEMO processor has validated a SAML assertion that does not contain the `<saml:DoNotCacheCondition>` element, the processor MAY continue to rely upon the assertion as long as the conditions of the assertion are satisfied, without re-validating the assertion at each instance of reliance. A NEMO processor that supports the `<saml:DoNotCacheCondition>` element MUST re-validate a SAML assertion containing a `<saml:DoNotCacheCondition>` element at each instance of reliance upon the SAML assertion.

Note: A SAML assertion containing the `<saml:DoNotCacheCondition>` element cannot be relied upon after the issuer's signing key has expired.

4.4.2 NEMO Role Attributes

This binding defines a NEMO node attribute called “roles”. The NEMO roles attribute SHALL be identified as a SAML attribute with namespace

```
&nemo;/attribute
```

and name

```
role
```

A NEMO node MAY be associated with zero or more roles. NEMO role attributes SHALL have associated values whose type is `xs:anyURI`. The `Issuer` and `Subject` of a NEMO role attribute statement MUST be NEMO nodes identified by their NEMO node identifiers.

4.4.3 NEMO Attribute Reliance Policy (Informative)

If the issuer of an attribute assertion is a NEMO node, other NEMO nodes may govern their reliance on the attribute assertion according to the roles held by the attribute assertion issuer.

4.5 *SSL/TLS Service Authorization*

Transport Layer Security [TLS] is a protocol for point-to-point message security. A NEMO client may simultaneously authenticate and authorize a service provider node by successfully establishing a TLS session with server-side authentication. Message integrity and confidentiality can be provided by the TLS transport, so services SHOULD NOT require higher-level message integrity and confidentiality protection when using TLS service authorization.

A service MUST indicate its support for TLS service authorization in its WSDL service description document. A service MUST also indicate in its service description document whether TLS client authentication is required (§5). NEMO clients using TLS service authorization are not required to support TLS client authentication.

Certificate authorities that issue certificates for the purpose of NEMO TLS service authorization SHALL issue certificates that comply with the X.509 profile specified by [WAPCertProf]. End entity certificates whose subjects are NEMO nodes SHALL include `Subject` fields containing the NEMO node's identifier, formatted as specified in §4.3.4.

Notes (informative):

1. TLS doesn't provide a way for a client to specify the certificate authorities trusted for server authentication. (TLS does provide a way for a server to specify the certificate authorities trusted for client authentication.) Servers supporting TLS authorization typically will be issued only a single certificate to be offered to clients. Clients must be configured with sufficient trust anchor certificates to authenticate all desired services.

2. TLS specifies authentication via X.509 certificates. While X.509 public key infrastructures are normally used only for binding certificate holders' names to their public keys, NEMO clients using TLS authorization may rely on certificate authorities to establish trust and authorization of service nodes.

3. NEMO does not specify the certification practices of certification authorities, or how clients manage certification trust anchors for TLS service authorization. In particular, NEMO does not specify whether some or all of the TLS service authorization trust anchors within a client can be managed or manipulated by device holders, or whether trust anchors are securely held away from device holders (tamper resistant).

4.5.1 Certificate Revocation

A NEMO client using TLS service authorization SHOULD NOT rely on or use any certificate identified in a valid certificate revocation list (CRL), as specified in [RFC3280].

4.5.2 Certificate and Key Renewal

NEMO nodes capable of secure time that support TLS service authorization MUST NOT use or accept expired certificates in TLS sessions. A NEMO node using TLS service authorization MUST NOT use or accept a certificate whose expiration date is after the issue date of a valid CRL received from the certificate authority that issued the certificate.

A certificate authority (CA) MUST establish a rollover period before the CA's certificate signing key expires. During the rollover period, the CA MUST issue certificates signed with the new certificate signing key, as well as a pair of rollover certificates certifying the expiring key with the new key, and certifying the new key with the expiring key. The rollover certificates MUST expire at the same time as the CA's older certificate signing key.

4.6 *Annex*

This annex includes all of the ASN.1 type and value definitions contained in this specification (§4) in the form of the ASN.1 module `NEMO`.

```
NEMO {iso(1) identified-organization(3) dod(6) internet(1) private(4)
enterprise(1) intertrust(7584) nemo(1)}
DEFINITIONS ::=
BEGIN

-- EXPORTS All --

IMPORTS

-- from Intertrust X.500 Object Identifier Specification
   id-itru
      FROM Intertrust {iso(1) identified-organization(3) dod(6)
      internet(1) private(4) enterprise(1) intertrust(7584)}

id-nemo OBJECT IDENTIFIER ::= {id-itru nemo(1)}
id-nat OBJECT IDENTIFIER ::= {id-nemo nameAttribute(1)}
id-xku OBJECT IDENTIFIER ::= {id-nemo extendedKeyUsage(2)}

uri ATTRIBUTE ::= {
   WITH SYNTAX UTF8String {ub-uri}
   EQUALITY MATCHING RULE caseExactMatch
   SUBSTRINGS MATCHING RULE caseExactSubstringsMatch
   ID id-nat-uri }

id-nat-uri OBJECT IDENTIFIER ::= {id-nat uri(1)}
ub-uri INTEGER ::= 1024

END -- NEMO
```

5 NEMO Policy Bindings

5.1 *Overview*

This section specifies bindings that can be used to express policies defining the security requirements for the NEMO Secure Messaging Protocol bindings. This section assumes a working knowledge of the following specifications:

NEMO Security Bindings (§3)

NEMO Trust Management Bindings (§4)

WS-Security [WS-SEC]

WS-SecureConversation [WS-SCON]

WS-Trust [WS-Trust]

WS-Policy [WS-POL]

WS-SecurityPolicy [WS-SEC-POL]

WS-PolicyAttachment [WS-POL-ATTCH]

Web Services Description Language 1.1 [WSDL 1.1]

Security Assertion Markup Language [SAML1.1]

5.2 *NEMO Web Service Policy Binding*

The NEMO Web Service Policy Binding defines a way for NEMO services to advertise their policies within a WSDL service description document [WSDL 1.1]. For a given service, these policies define the particular protocol options followed to establish a secure message channel, as well as the service's authorization requirements and guarantees.

Policies SHOULD make reference to the protocols observed by a service, and MUST NOT conflict with any choices specified in the protocol definition. Any such conflicting policy elements SHOULD be ignored and MUST NOT be enforced. If a service description excludes policy specifications for options available in the protocol binding specification, or if an assertion is explicitly declared optional, then any permitted option MAY be used.

For example, a service's policy implementing a Web Service binding of the Basic Secure Messaging Protocol defined in (§3) may specify whether confidentiality is required or rejected, and must not override the encryption algorithm choices defined by the binding. If the service's policy does not specify confidentiality protection policy, then confidentiality protection is optional, per the Basic Secure Messaging Protocol.

Policies MAY use policy assertions other than those defined in this binding to express policies, and MAY refine policy scope, so long as the policies do not conflict with the protocol definition.

WS-PolicyAttachment [WS-POL-ATTCH] may be used to provide scope for policies. Unless policies are explicitly attached to particular scopes, they are assumed to be in-scope for all communication where they are applicable.

The usages defined in [WS-POL] SHALL be used inside the policy assertions to declare protocol option selections.

5.2.1 Protocol Policy

A protocol may be referenced by its identifier. The protocol is refined using policies, which can be attached to the protocol using WS-PolicyAttachment [WS-POL-ATTCH].

5.2.1.1 Security Tokens

A `<wssp:SecurityToken>` element may assert the presence of a token named in the protocol. The attribute `nemosec:Usage` defined in §3 SHALL be used to signal the usage of the token to which the `<wssp:SecurityToken>` element corresponds.

5.2.1.2 nemop:Usage Attribute

The nemop:Usage attribute MAY be included within policy assertions to indicate that the assertion has a distinguished role in a certain context, such as a communications protocol. NEMO message processors can use the `nemop:Usage` attribute as a hint to locate policy assertions relating to specific contexts.

5.2.1.3 nemosec:Usage Attribute

The `nemosec:Usage` attribute MAY be used within a `<wssp:SecurityToken>` element to identify a policy assertion that indicates requirements or capabilities related to a token with a distinguished role in a certain context, such as a communications protocol. NEMO message processors can use the `nemosec:Usage` attribute within a `<wssp:SecurityToken>` element as a hint to locate message tokens relating to specific contexts.

5.2.1.4 MessageAge

Whenever timestamp support is defined as optional by the protocol, the policy MAY specify whether the timestamp is used in the particular interaction, and MAY specify what maximum message age is accepted. A `<wssp:MessageAge>` assertion SHALL be used to signal message timestamp usage.

A policy assertion specifying the use of an optional timestamp within the following protocols (§3) MAY contain the attribute `nemop:Usage` with the value indicated below.

Protocol	Attribute Value
NEMO Basic Secure Messaging	`&nemosec;/secure-protocol/basic/1.0/policyAssertion#timestamp`
NEMO Secure Conversation Protocol	`&nemosec;/secure-protocol/secure-conversation/1.0/policyAssertion#timestamp`

5.2.1.5 Nonce

Whenever nonce usage is defined as optional by the protocol, the policy MAY specify whether a nonce is used in the the particular interaction. The element `<nemop:Nonce>` SHALL be used to signal whether a nonce is used by the service.

The syntax for the `<nemop:Nonce>` element is

.../Nonce

This element is a policy assertion expressing the requirement for a nonce in a message.

.../{any}

This is an extensibility mechanism allowing other elements describing the nonce.

.../@{any}

This is an extensibility mechanism allowing attributes describing the nonce.

A policy assertion specifying the use of an optional nonce within the following protocols (§3) MAY contain the attribute `nemop:Usage` with the value indicated below.

Protocol	Attribute Value
NEMO Basic Secure Messaging	`&nemosec;/secure-protocol/basic/1.0/policyAssertion#nonce`
NEMO Secure Conversation Protocol	`&nemosec;/secure-protocol/secure-conversation/1.0/policyAssertion#nonce`

5.2.1.6 Signed Message Key

Whenever a protocol defines as optional that the secret key used to encrypt the message contents be signed, the policy MAY specify whether the signed message key is required to be present in the message. It is RECOMMENDED that this assertion also be explicitly specified in the policy when the protocol requires the secret key to be signed conditioned on both integrity and confidentiality being present. A `<wssp:SecurityToken>` element with `TokenType` of `&nemosec;/SymmetricKey` SHALL be used to signal whether the symmetric key is required (see §5.2.1.1).

5.2.1.7 Signed Challenge / Signed Challenge Response

Whenever a signed challenge/response usage is defined as optional by the protocol, the policy MAY specify whether signed challenge/response is used in the particular interaction. The elements `<nemop:SignChallenge>` and `<nemop:SignChallengeResponse>` SHALL be used to signal whether signed challenge and response are used by the service.

The syntax for the `<nemop:SignChallenge>` element is

.../SignChallenge

This element is a policy assertion expressing the requirement for a signed challenge in a message.

.../{any}

This is an extensibility mechanism allowing other elements describing the signed challenge.

.../@{any}

This is an extensibility mechanism allowing attributes describing the signed challenge.

The syntax for the `<nemop:SignChallengeResponse>` element is

.../SignChallengeResponse

This element is a policy assertion expressing the requirement for a signed challenge response in a message.

.../{any}

This is an extensibility mechanism allowing other elements describing the signed challenge response.

.../@{any}

This is an extensibility mechanism allowing attributes describing the signed challenge response.

A policy assertion specifying the use of an optional signed challenge within the following protocols (§3) MAY contain the attribute `nemop:Usage` with the value indicated below.

Protocol	Attribute Value
NEMO Basic Secure Messaging	`&nemosec;/secure-protocol/basic/1.0/policyAssertion#signChallenge`
NEMO Secure Conversation Protocol	`&nemosec;/secure-protocol/secure-conversation/1.0/policyAssertion#signChallenge`

5.2.1.8 Message Integrity

Whenever message integrity usage is defined as optional by the protocol, the policy MAY specify whether message integrity is used in the the particular interaction. The element `<wssp:Integrity>` SHALL be used to signal whether message integrity is used by the service.

A policy assertion specifying the optional use of integrity protection within the following protocols (§3) MAY contain the attribute `nemop:Usage` with the value indicated below.

Protocol	Attribute Value
NEMO Basic Secure Messaging	`&nemosec;/secure-protocol/basic/1.0/policyAssertion#integrity`
NEMO Secure Conversation Protocol	`&nemosec;/secure-protocol/secure-conversation/1.0/policyAssertion#integrity`

5.2.1.9 Message Confidentiality

Whenever message confidentiality usage is defined as optional by the protocol, the policy MAY specify whether message confidentiality is used in the particular interaction. A `<wssp:Confidentiality>` element SHALL be used to signal whether message confidentiality is used by the service.

A policy assertion specifying the optional use of confidentiality protection within the following protocols (§3) MAY contain the attribute `nemop:Usage` with the value indicated below.

Protocol	Attribute Value
NEMO Basic Secure Messaging	`&nemosec;/secure-protocol/basic/1.0/policyAssertion#confidentiality`
NEMO Secure Conversation Protocol	`&nemosec;/secure-protocol/secure-conversation/1.0/policyAssertion#confidentiality`

5.2.1.10 Message Parts

This specification defines a dialect of the `wssp:MessageParts` mechanism defined in [WS-SEC-POL]. This dialect extends, and thereby includes all the message part functions defined in, the dialect specified in [WS-POL-ASSRT], identified by the URI

```
http://schemas.xmlsoap.org/2002/12/wsse#part
```

/wssp:MessageParts/@Dialect

> When using the dialect specified here, the `@Dialect` attribute should be the URI
>
> `http://nemo.intertrust.com/2004/policy#part`

/wssp:MessageParts

> When using the mechanism specified here, the contents of the `<wssp:MessageParts>` element is a string of the form
>
> `nemop:Token(usage)`
>
> where “usage” represents a URI indicating the usage of the message element in the messaging protocol.

Note: The attribute `nemosec:Usage` defined in §3 MAY be attached to message elements as a hint to receiving processors trying to locate message elements with a given protocol usage. Receiving processors are nonetheless ultimately responsible for locating and verifying the usage of elements that have usages within the semantics of a defined protocol.

5.2.1.11 XML Element Encryption Policy

According to [WS-SEC], the tags of an `<S11:Body>` element SHALL NOT be encrypted. A policy requiring the encryption of the `<S11:Body>` element indicates one of three methods for encrypting the SOAP Body.

- A processor MAY encrypt the `<S11:Body>` element according to [XMLENC], by replacing the contents of the `<S11:Body>` element with an `<enc:EncryptedData>` element with a `Type` attribute having the value `http://www.w3.org/2001/04/xmlenc#Content`

  This is the RECOMMENDED method of encrypting an `<S11:Body>` element that has content. This method MAY be applied even if the unencrypted `<S11:Body>` element has no content. In this case, the `<enc:EncryptedData>` element’s cipher data SHALL resolve to an empty octet sequence.

- If the unencrypted `<S11:Body>` element contains only whitespace and a single child element, then a processor MAY satisfy the encryption policy by encrypting the child element, as specified in [XMLENC]. In this case, the entirety of the child element must be encrypted, including the child element's tags.

- If the unencrypted `<S11:Body>` element has no content, then a processor MAY satisfy the encryption policy by not altering the empty `<S11:Body>` element. This is the RECOMMENDED method of encrypting an `<S11:Body>` element that has no content.

According to [WS-SEC], the `<S11:Header>` and `<S11:Envelope>` elements of a SOAP message SHALL NOT be encrypted. To satisfy policy requiring the encryption of a message element that is not an `<S11:Body>`, `<S11:Header>` or `<S11:Envelope>` element, a processor SHALL encrypt the entire element, including the element tags, as specified in [XMLENC].

5.2.2 Message Security Policy

The Message Security Policy governs message-level security between a client and a service. In particular, message security policy MAY require that a particular message protocol definition be observed.

Message Security policy MAY be attached to the WSDL components, as described in Section 4 of the WS-PolicyAttachment specification [WS-POL-ATTCH].

5.2.2.1 Protocol Assertion

A policy MAY make reference to the protocol that an operation implements, or the protocol step that a message represents. A `<nemop:ProtocolAssertion>` element SHALL be used to specify a particular protocol or protocol step.

The syntax for `<nemop:ProtocolAssertion>` is as follows:

```
<ProtocolAssertion wsu:Id=”...”?>
   <Reference URI=”...”/>?
   <nemosec:Step index=”…” type=”…”/>?
</ProtocolAssertion>
```

The following describes the elements defined above.

/ProtocolAssertion

This contains the protocol assertion.

/ProtocolAssertion/Reference

This element contains a URI identifying the protocol definition that is asserted by this policy assertion.

/ProtocolAssertion/Reference/@URI

This attribute specifies a URI identifying the protocol definition.

/ProtocolAssertion/nemosec:Step

This element MAY be used when the `ProtocolAssertion` is attached to a message. The `<nemosec:Step>` element indicates the step in a protocol that the message represents. See §3 for a specification of the children of this element.

/ProtocolAssertion/@{any}

This is an extensibility mechanism to allow additional attributes, based on schemas, to be added.

/ProtocolAssertion/{any}

This is an extensibility mechanism to allow different (extensible) types of security information, based on a schema, to be passed.

5.2.2.2 Profile Assertion

The Profile policy assertion applies to operations. The assertion references a NEMO profile that the service operation implements. A `<nemosec:Profile>` element defined in §3 SHALL be used within a `<wsp:Policy>` element to assert conformance to a particular NEMO profile.

5.2.3 Application Security Policy

In addition to the Message Security Policy, the service MAY publish its application policies. The policies can be described using the same elements as defined in §5.2.2, as well as other policy elements. In particular, SAML [SAML1.1] Security Token assertions may be used to express application policy assertions.

Application Security policy MAY be attached to the WSDL components, as described in Section 4 of the WS-PolicyAttachment specification [WS-POL-ATTCH].

5.2.3.1 SAML Attribute Assertion Token

Whenever a SAML attribute assertion token [SAML1.1] is used by the service, the service policy MAY signal attributes and attribute values associated with the requestor NEMO node. This SHALL be expressed using a `<SecurityToken>` element declared in the scope of the service's policy with a `TokenType` of `wssp:SAMLAssertion`, as specified in WS-SecurityPolicy [WS-SEC-POL].

When the `wssp:SecurityToken/wssp:TokenType` is `wssp:SAMLAssertion`, the `wssp:SecurityToken/wssp:Claims` element MAY contain a set of `<nemop:NameValuePairDescription>` elements. The `<wssp:SecurityToken>` policy assertion describes the valid assertion and presentation of SAML attribute statement(s). SAML attributes' names and values SHALL be described via the `<nemop:NameValuePairDescription>` element. The `<wssp:SecurityToken>` element's asserted policy is the association of the subject NEMO node with the indicated attributes and values, as described in NEMO Trust Management Bindings (§4).

If the `SecurityToken/Claims` element contains more than one child element, then the effect of the `SecurityToken` policy assertion is the same as the effect of a `<wsp:All>` element containing several `SecurityToken` policy assertions, each with one of the child elements within the `SecurityToken/Claims` element.

/wssp:SecurityToken/wssp:Claims/NameValuePairDescription

This element identifies a node attribute and optionally values to be associated with the subject NEMO node.

/wssp:SecurityToken/wssp:Claims/NameValuePairDescription/@Name

The name of the NEMO node attribute.

/wssp:SecurityToken/wssp:Claims/NameValuePairDescription/@Namespace

The namespace of the NEMO node attribute.

/wssp:SecurityToken/wssp:Claims/NameValuePairDescription/ValuePattern

This element contains a value pattern. Each `ValuePattern` in a `NameValuePairDescription` must match an attribute value associated with the subject NEMO node and the indicated attribute.

Note: It is possible for two distinct `<nemop:ValuePattern>` elements to be satisfied by a single value, if the `<nemop:ValuePattern>` elements signal overlapping patterns.

/wssp:SecurityToken/wssp:Claims/NameValuePairDescription/ValuePattern/@MatchType

This optional string attribute determines how to match a NEMO node attribute value with the contents of the `<nemop:ValuePattern>` element. This binding defines the value

```
wssp:Exact
```

to signal that the `<nemop:ValuePattern>` element matches only its contents, subject to a canonicalization algorithm.

The default value for this attribute is wssp:Exact.

/wssp:SecurityToken/wssp:Claims/NameValuePairDescription/ValuePattern/@Canonicalization

This optional URI attribute signals the canonicalization algorithm used to match XML data to the contents of the ValuePattern. The default canonicalization algorithm for element content is XML Exclusive Canonicalization [EXC-C14N]. The default canonicalization algorithm for character data content conforms to the text node serialization specified in [EXC-C14N].

The processing rules specified by the NEMO Trust Management Bindings (§4) SHALL apply when evaluating SAML assertions with regard to policies. Consequently, the following claims inside a SAML Security Token assertion are semantically equivalent:

1. Multiple `<nemop:NameValuePairDescription>` elements with the same attribute name.

```
<wssp:Claims xmlns:example=”http://www.example.com/attribute”>
     <nemop:NameValuePairDescription
          Name=”attributeName”
          Namespace=”http://www.example.com/attribute”>
        <nemop:ValuePattern>
          <example:AAA/>
        </nemop:ValuePattern>
     </nemop:NameValuePairDescription>

     <nemop:NameValuePairDescription
          Name=”attributeName”
          Namespace=”http://www.example.com/attribute”>
        <nemop:ValuePattern>
          <example:BBB/>
        </nemop:ValuePattern>
     </nemop:NameValuePairDescription>
</wssp:Claims>
```

2. A single `<nemop:NameValuePairDescription>` element enclosing multiple `<nemop:ValuePattern>` elements

```
<wssp:Claims xmlns:example=”http://www.example.com/attribute”>
     <nemop:NameValuePairDescription
          Name=”attributeName”
          Namespace=”http://www.example.com/attribute”>
        <nemop:ValuePattern>
          <example:AAA/>
        </nemop:ValuePattern>
        <nemop:ValuePattern>
          <example:BBB/>
        </nemop:ValuePattern>
     </nemop:NameValuePairDescription>
</wssp:Claims>
```

5.2.4 Session Policy

Whenever the service supports the Secure Conversation Binding specified in §3, the Session Service MAY publish policies for establishing the session. Messaging requirements SHALL be specified as described in §5.2.2. Additionally, the following policies MAY be expressed.

5.2.4.1 Session Duration

Session policy SHOULD specify the session’s minimum and maximum lifetime. This SHALL be expressed using the `<nemop:SessionDuration>` element declared in the scope of the Session Service of the service node.

The syntax for this element is as follows:

```
   <nemop:SessionDuration>
      <nemop:MinimumDuration>
      ...
      </nemop:MinimumDuration>

      <nemop:MaximumDuration>
      ...
      </nemop:MaximumDuration>
   </nemop:SessionDuration>
```

The following describes the attributes and elements listed in the schema overview above:

/SessionDuration

This is the policy element specifying the range of session durations for the sessions that the Session Service can generate.

/SessionDuration/MiniumDuration

This is the element specifying the minimum session duration. This element is of the same schema type as the `<wssp:MessageAge>` element [WS-SEC-POL].

/SessionDuration/MaximumDuration

This is the element specifying the maximum session duration. This element is of the same schema type as the `<wssp:MessageAge>` element [WS-SEC-POL].

/SessionDuration/@{any}

This is an extensibility mechanism to allow additional attributes, based on schemas, to be added.

/SessionDuration/{any}

This is an extensibility mechanism to allow different (extensible) ways of specifying session duration policy, based on a schema, to be passed. Unrecognized elements SHOULD be ignored.

5.2.5 Caching Policy

Policies for services that support sessions MAY include policies for caching the validity of credentials that the client node has previously provided to the service node within the current session. If a credential validity is cached, the client node does not need to resubmit the credential within the session. (There may be cases when credential validity expires prematurely—for example, if a credential is revoked.)

If the client resubmits a credential whose validity has been previously cached within the current session, the service MAY revalidate the credential and re-cache it, or the service MAY ignore the supplied credential and keep the current cache state.

The service node MAY choose to cache credential validity for longer than the session. This specification doesn't provide a mechanism to signal that behavior.

5.2.5.1 Signaling Credential Caching

Policy MAY signal whether the service node caches the validity of credentials within sessions. The policy SHALL be expressed using the `nemop:Cache` attribute, whose values may be "true" or "false". If the `nemop:Cache` attribute is missing, its default value is "false". If a needed credential is not cached by the service node, clients SHOULD supply the credential with each request.

Syntax:

```
<...AnyCredentialPolicy nemop:Cache=”...”/>
```

5.2.6 Examples

5.2.6.1 Basic Protocol – No Sessions

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="LicenseManager"
targetNamespace="http://example.com/myservice"
```

```
xmlns:tns="http://example.com/myservice "
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsp="http://schemas.xmlsoap.org/ws/2003/12/policy"
xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wsswssecurity-secext-1.0.xsd"
xmlns:wssp="http://schemas.xmlsoap.org/ws/2002/12/secext/"
xmlns:saml=”urn:oasis:names:tc:SAML:1.0:assertion”
xmlns:nemop=”http://nemo.intertrust.com/2004/policy”
xmlns:nemosec=”http://nemo.intertrust.com/2005/10/security”>
<wsp:UsingPolicy wsdl:Required="true" />

   <!-- Service Node’s Public Encryption Key -->
   <wsse:SecurityTokenReference
      nemosec:Usage=”&nemosec;/secure-protocol/basic/1.0#request-
encryptionKey”>
      <wsse:Embedded>
         <wsse:BinarySecurityToken
            ValueType=”http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-x509-token-profile-1.0#X509PKIPathv1”
            EncodingType=”wsse:Base64Binary”>
                  ...X509 Certificate...
         </wsse:BinarySecurityToken>
      </wsse:Embedded>
   </wsse:SecurityTokenReference>

   <!-- Service Node’s Public Signing Key -->
   <wsse:SecurityTokenReference
      nemosec:Usage=”&nemosec;/secure-protocol/basic/1.0#response-
signingKey”>
      <wsse:Embedded>
         <wsse:BinarySecurityToken
            ValueType=”http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-x509-token-profile-1.0#X509PKIPathv1”
            EncodingType=”wsse:Base64Binary”>
                  ...X509 Certificate...
         </wsse:BinarySecurityToken>
      </wsse:Embedded>
   </wsse:SecurityTokenReference>

   <!—Service’s role assertion -->
   <wsse:SecurityTokenReference
      nemosec:Usage=”&nemo;/attribute/role”>
      <wsse:Embedded>
         <saml:Assertion
            AssertionID=”…”
            IssueInstant=”…”
            Issuer=”urn:nemo:node:CA”
            MajorVersion=”1”
            MinorVersion=”1”>
            <saml:AttributeStatement>
              <saml:Subject>
```

```
               <saml:NameIdentifier>
                    uri=urn:nemo:node:A
               </saml:NameIdentifier>
          </saml:Subject>
          <saml:Attribute
               AttributeName=”role”
               AttributeNamespace=”…/attribute”>
               <saml:AttributeValue>
                    MyServiceRole
               </saml:AttributeValue>
          </saml:Attribute>
         </saml:AttributeStatement>
         <ds:Signature>
         </ds:Signature>
      </saml:Assertion>
   </wsse:Embedded>
</wsse:SecurityTokenReference>

<types>
   <xsd:schema>
      <xsd:complexType name=”RequestType”/>
      <xsd:complexType name=”ResponseType”/>
      <xsd:element name=”RequestPayload” type=”RequestType”/>
      <xsd:element name=”ResponsePayload”
                    type=”ResponseType”/>
   </xsd:schema>
</types>

<message name=”Request”>
   <part name=”Request” element=”RequestPayload”/>
</message>

<message name=”Response”>
   <part name=”Response” element=”ResponsePayload”/>
</message>

<portType name=”MyPortType”>
   <operation name=”MyOperation”>
      <input name=”Request” message=”Request”/>
      <output name=”Response” message=”Response”/>
   </operation>
</portType>

<binding name=”MyPortTypeSoapBinding” type=”MyPortType”>
   <wsdlsoap:binding style=”document” transport=”…soap/http”/>
   <operation name=”MyOperation”>
      <wsp:PolicyReference URI=”#MyOperationPolicy”/>
      <wsdlsoap:operation/>
      <input>
         <wsp:PolicyReference URI=”#RequestPolicy”/>
         <wsdlsoap:body use=”literal” namespace=”…”/>
      </input>
```

```
         <output>
            <wsp:PolicyReference URI=”#ResponsePolicy”/>
            <wsdlsoap:body use=”literal” namespace=”…”/>
         </output>
      </operation>
   </binding>

   <service name=”MyService”>
      <port name=”MyPort” binding=”MyPortTypeSoapBinding”>
         <wsdlsoap:Address location=”http://...”/>
      </port>
   </service>

   <!–MyOperation Policy -->
   <wsp:Policy wsu:Id=”MyOperationPolicy”>
      <!–Profile -->
      <nemosec:Profile URI=”&nemo;/profile/main”/>
      <!–Protocol -->
      <nemop:ProtocolAssertion>
         <nemop:Reference URI=”&nemosec;/secure-
            protocol/basic/1.0”/>
      </nemop:ProtocolAssertion>
   </wsp:Policy>

   <!–Request Policy -->
   <wsp:Policy wsu:Id=”RequestPolicy”>

      <!–Protocol -->
      <nemop:ProtocolAssertion>
         <nemop:Reference URI=”&nemosec;/secure-
            protocol/basic/1.0”/>
         <nemosec:Step type=”request”/>
      </nemop:ProtocolAssertion>

      <!–Client’s encryption key -->
      <wssp:SecurityToken
         nemosec:Usage=”&nemosec;/secure-
            protocol/basic/1.0#response-encryptionKey”/>

      <nemop:Nonce
         nemop:Usage=”&nemosec;/secure-
            protocol/basic/1.0/policyAssertion#nonce”/>
      <wssp:MessageAge Age="3600"
         nemop:Usage=”&nemosec;/secure-
            protocol/basic/1.0/policyAssertion#timestamp”/>

      <!–client’s signature key -->
      <wssp:SecurityToken
            nemosec:Usage=”&nemosec;/secure-
              protocol/basic/1.0#request-signingKey”>
         <wsse:TokenType>http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-x509-token-profile-1.0#X509PKIPathv1</wssp:TokenType>
```

```
         <wssp:TokenIssuer>
           ...Trusted Roots...
         </wssp:TokenIssuer>
      </wssp:SecurityToken>

      <!— The keys to be used for confidentiality and integrity -->
      <!-  are specified by the protocol binding -->
      <wssp:Confidentiality
         nemop:Usage=”&nemosec;/secure-
            protocol/basic/1.0/policyAssertion#confidentiality”>
         <wssp:MessageParts Dialect=”&nemop;#part”>
            nemop:Token(&nemosec;/secure-
              protocol/basic/1.0#request-messageKey)
            wsp:Body()
         </wssp:MessageParts>
      </wssp:Confidentiality>
      <wssp:Integrity
         nemop:Usage=”&nemosec;/secure-
            protocol/basic/1.0/policyAssertion#integrity”>
         <wssp:MessageParts Dialect=”&nemop;#part”>
            nemop:Token(&nemosec;/secure-
              protocol/basic/1.0#request-messageKey)
            wsp:Body()
            nemop:Token(&nemosec;/…/request-timestamp)
            nemop:Token(&nemosec;/…/request-nonce)
            nemop:Token(&nemosec;/…/request-toNode)
         </wssp:MessageParts>
      </wssp:Integrity>

      <!-- client’s Role (application policy attribute token) -->
      <wsp:ExactlyOne>
         <wssp:SecurityToken>
            <wssp:TokenType>wsse:SAMLAssertion</wssp:TokenType>
            <wssp:TokenIssuer>...Trusted Roots...
            </wssp:TokenIssuer>
            <wssp:Claims>
              <nemop:NameValuePairDescription
                  Name=”role”
                  Namespace=”&nemop;/attribute”>
                  <nemop:ValuePattern>
                      MyClientRole
                  </nemop:ValuePattern>
              </nemop:NameValuePairDescription>
            </wssp:Claims>
         </wssp:SecurityToken>

         <wssp:SecurityToken>
            <wssp:TokenType>wsse:SAMLAssertion</wssp:TokenType>
            <wssp:TokenIssuer>...Trusted Roots...
            </wssp:TokenIssuer>
            <wssp:Claims>
              <nemop:NameValuePairDescription
```

```
                  Name=”role”
                  Namespace=”&nemop;/attribute”>
                  <nemop:ValuePattern>
                      MyAlternateClientRole
                  </nemop:ValuePattern>
              </nemop:NameValuePairDescription>
            </wssp:Claims>
         </wssp:SecurityToken>
      </wsp:ExactlyOne>
   </wsp:Policy>

   <!–Response Policy -->
   <wsp:Policy wsu:Id=”ResponsePolicy”>

      <!–Protocol -->
      <nemop:ProtocolAssertion>
         <nemop:Reference URI=”&nemosec;/secure-
            protocol/basic/1.0”/>
         <nemosec:Step type=”response”/>
      </nemop:ProtocolAssertion>

      <!–client’s encryption key -->
      <wssp:SecurityToken
         nemosec:Usage=”&nemosec;/secure-
            protocol/basic/1.0#response-encryptionKey”>
         <wsse:TokenType>http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-x509-token-profile-1.0#X509PKIPathv1</wssp:TokenType>
         <wssp:TokenIssuer>
            ...Trusted Roots...
         </wssp:TokenIssuer>
      </wssp:SecurityToken>

      <!–client’s signature key -->
      <wssp:SecurityToken
         nemosec:Usage=”&nemosec;/secure-
            protocol/basic/1.0#request-signingKey”/>

      <nemop:Nonce
         nemop:Usage=”&nemosec;/secure-
            protocol/basic/1.0/policyAssertion#nonce”/>
      <wssp:MessageAge Age="3600"
         nemop:Usage=”&nemosec;/secure-
            protocol/basic/1.0/policyAssertion#timestamp”/>
      <wssp:Confidentiality
         nemop:Usage=”&nemosec;/secure-
            protocol/basic/1.0/policyAssertion#confidentiality”>
         <wssp:MessageParts Dialect=”&nemop;#part”>
            nemop:Token(&nemosec;/secure-
              protocol/basic/1.0#response-messageKey)
            wsp:Body()
         </wssp:MessageParts>
      </wssp:Confidentiality>
      <wssp:Integrity
         nemop:Usage=”&nemosec;/secure-
            protocol/basic/1.0/policyAssertion#integrity”>
         <wssp:MessageParts Dialect=”&nemop;#part”>
            nemop:Token(&nemosec;/secure-
              protocol/basic/1.0#response-messageKey)
            wsp:Body()
            nemop:Token(&nemosec;/…/response-timestamp)
            nemop:Token(&nemosec;/…/response-nonce)
            nemop:Token(&nemosec;/…/response-toNode)
         </wssp:MessageParts>
      </wssp:Integrity>
   </wsp:Policy>
   ...
</wsdl:definitions>
```

5.2.6.2 Secure Conversation Protocol

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="SessionService"
          targetNamespace="http://example.com/sessionservice "
          xmlns:tns="http://example.com/sessionservice "
          xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
          xmlns="http://schemas.xmlsoap.org/wsdl/"
          xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
          xmlns:wsp=
                 "http://schemas.xmlsoap.org/ws/2003/12/policy"
          xmlns:wsse=
                  "http://docs.oasis-open.org/wss/2004/01/oasis-
            200401-wsswssecurity-secext-1.0.xsd">
          xmlns:wssp=
                  "http://schemas.xmlsoap.org/ws/2002/12/secext"
          xmlns:nemop="http://nemo.intertrust.com/2004/policy”
         xmlns:nemosec=http://nemo.intertrust.com/2005/10/security>

    <wsp:UsingPolicy wsdl:Required="true" />

   <!-- Service Node’s Public Encryption Key -->
   <wsse:SecurityTokenReference
        nemosec:Usage=”&nemosec;/secure-protocol/secure-
            conversation/1.0#establishment-request-
            encryptionKey”>
         <wsse:Embedded>
           <wsse:BinarySecurityToken
             ValueType=”http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-x509-token-profile-1.0#X509PKIPathv1”
             EncodingType=”wsse:Base64Binary”>
                    ...X509 Certificate...
           </wsse:BinarySecurityToken>
         </wsse:Embedded>
   </wsse:SecurityTokenReference>

   <!-- Service Node’s Public Signing Key -->
   <wsse:SecurityTokenReference
        nemosec:Usage=”&nemosec;/secure-protocol/secure-
            conversation/1.0#establishment-response-signingKey”>
         <wsse:Embedded>
           <wsse:BinarySecurityToken
             ValueType=”http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-x509-token-profile-1.0#X509PKIPathv1”
             EncodingType=”wsse:Base64Binary”>
                    ...X509 Certificate...
           </wsse:BinarySecurityToken>
         </wsse:Embedded>
   </wsse:SecurityTokenReference>

   <!–Service’s role assertion -->
   <wsse:SecurityTokenReference
      nemosec:Usage=”&nemo;/attribute/role”>
      <wsse:Embedded>
         <saml:Assertion
            AssertionID=”…”
            IssueInstant=”…”
            Issuer=”urn:nemo:node:CA”
            MajorVersion=”1”
            MinorVersion=”1”>
            <saml:AttributeStatement>
              <saml:Subject>
                  <saml:NameIdentifier>
                      uri=urn:nemo:node:A
                  </saml:NameIdentifier>
              </saml:Subject>
              <saml:Attribute
                  AttributeName=”role”
                  AttributeNamespace=”…/attribute”>
                  <saml:AttributeValue>
                      MyServiceRole
                  </saml:AttributeValue>
              </saml:Attribute>
            </saml:AttributeStatement>
            <ds:Signature>
            </ds:Signature>
         </saml:Assertion>
      </wsse:Embedded>
   </wsse:SecurityTokenReference>

   <types>
      <xsd:schema>
         <xsd:import namespace=”&nemosec;”/>
         <xsd:complexType name=”RequestType”/>
         <xsd:complexType name=”ResponseType”/>
         <xsd:import namespace=”http://.../trust”/>
        <xsd:import namespace=http://.../sc”/>
      </xsd:schema>
   </types>

   <message name=”SessionRequest”>
      <part name=”Request” element=”wst:RequestSecurityToken”/>
   </message>

   <message name=”SessionResponse”>
      <part name=”Response”
         element=”wst:RequestSecurityTokenResponse”/>
   </message>

   <message name=”Request”>
      <part name=”Request” type=”RequestPayloadType”/>
   </message>

   <message name=”Response”>
      <part name=”Response” type=”ResponsePayloadType”/>
   </message>

   <portType name=”MyPortType”>
      <operation name=”MyOperation”>
         <input name=”Request” message=”Request”/>
         <output name=”Response” message=”Response”/>
      </operation>
   </portType>

   <portType name=”SessionPortType”>
      <operation name=”EstablishSession”>
         <input name=”Request” message=”SessionRequest”/>
         <output name=”Response” message=”SessionResponse”/>
      </operation>
   </portType>

   <binding name=”MyPortTypeSoapBinding” type=”MyPortType”>
      <wsdlsoap:binding style=”document” transport=”…soap/http”/>
      <operation name=”MyOperation”>
         <wsp:PolicyReference URI=”#MyOperationPolicy”/>
         <wsdlsoap:operation/>
         <input>
            <wsp:PolicyReference URI=”#RequestPolicy”/>
            <wsdlsoap:body use=”literal” namespace=”…”/>
         </input>
         <output>
            <wsp:PolicyReference URI=”#ResponsePolicy”/>
            <wsdlsoap:body use=”literal” namespace=”…”/>
         </output>
      </operation>
   </binding>

   <binding name=”SessionPortTypeSoapBinding”
```

```
      type=”SessionPortType”>
      <wsdlsoap:binding style=”document” transport=”…soap/http”/>
      <operation name=”EstablishSession”>
         <wsp:PolicyReference URI=”#EstablishSessionPolicy”/>
         <wsdlsoap:operation/>
         <input>
            <wsp:PolicyReference URI=”#SessionRequestPolicy”/>
            <wsdlsoap:body use=”literal” namespace=”…”/>
         </input>
         <output>
            <wsp:PolicyReference URI=”#SessionResponsePolicy”/>
            <wsdlsoap:body use=”literal” namespace=”…”/>
         </output>
      </operation>
   </binding>

   <service name=”MyService”>
      <port name=”MyPort” binding=”MyPortTypeSoapBinding”>
         <wsdlsoap:address location=”http://...”/>
      </port>
   </service>

   <service name=”SessionService”>
      <port name=”SessionPort”
         binding=”SessionPortTypeSoapBinding”>
         <wsdlsoap:Address location=”http://...”/>
      </port>
   </service>

   <!—MyOperation Policy -->
   <wsp:Policy wsu:Id=”MyOperationPolicy”>
      <!—Profile -->
      <nemosec:Profile URI=”&nemo;/profile/main”/>
      <!—Protocol -->
      <nemop:ProtocolAssertion>
         <nemop:Reference URI=”&nemosec;/secure-protocol/secure-
            conversation/1.0”/>
      </nemop:ProtocolAssertion>
   </wsp:Policy>

   <!— Request Policy -->
   <wsp:Policy wsu:Id=”RequestPolicy”>
      <!—Protocol -->
      <nemop:ProtocolAssertion>
         <nemop:Reference URI=”&nemosec;/secure-protocol/secure-
            conversation/1.0”/>
         <nemosec:Step type=”request”>
      </nemop:ProtocolAssertion>
      <nemop:Nonce
         nemop:Usage=”&nemosec;/secure-
            protocol/secure-
            conversation/1.0/policyAssertion#nonce”/>
      <wssp:MessageAge Age="3600"
         nemop:Usage=”&nemosec;/secure-
            protocol/secure-
            conversation/1.0/policyAssertion#timestamp”/>
      <wssp:Confidentiality
         nemop:Usage=”&nemosec;/secure-
            protocol/secure-
            conversation/1.0/policyAssertion#confidentiality”>
         <wssp:MessageParts Dialect=”&nemop;#part”>
            wsp:Body()
         </wssp:MessageParts>
      </wssp:Confidentiality>
      <wssp:Integrity
         nemop:Usage=”&nemosec;/secure-
            protocol/secure-
            conversation/1.0/policyAssertion#integrity”>
            wsp:Body()
            nemop:Token(&nemosec;/…/request-timestamp)
            nemop:Token(&nemosec;/…/request-nonce)
            nemop:Token(&nemosec;/…/request-toNode)
         </wssp:MessageParts>
      </wssp:Integrity>
   </wsp:Policy>

   <!– Response Policy -->
   <wsp:Policy wsu:Id=”ResponsePolicy”>
      <!–Protocol -->
      <nemop:ProtocolAssertion>
         <nemop:Reference URI=”&nemosec;/secure-protocol/secure-
            conversation/1.0”/>
         <nemosec:Step type=”response”>
      </nemop:ProtocolAssertion>
      <nemop:Nonce
         nemop:Usage=”&nemosec;/secure-
            protocol/secure-
            conversation/1.0/policyAssertion#nonce”/>
      <wssp:MessageAge Age="3600"
         nemop:Usage=”&nemosec;/secure-
            protocol/secure-
            conversation/1.0/policyAssertion#timestamp”/>
      <wssp:Confidentiality
         nemop:Usage=”&nemosec;/secure-
            protocol/secure-
            conversation/1.0/policyAssertion#confidentiality”>
         <wssp:MessageParts Dialect=”&nemop;#part”>
            wsp:Body()
         </wssp:MessageParts>
      </wssp:Confidentiality>
      <wssp:Integrity
         nemop:Usage=”&nemosec;/secure-
            protocol/secure-
            conversation/1.0/policyAssertion#integrity”>
         <wssp:MessageParts Dialect=”&nemop;#part”>
            wsp:Body()
            nemop:Token(&nemosec;/…/response-timestamp)
            nemop:Token(&nemosec;/…/response-nonce)
            nemop:Token(&nemosec;/…/response-toNode)
         </wssp:MessageParts>
      </wssp:Integrity>
   </wsp:Policy>

   <!—Establish Session Policy -->
   <wsp:Policy wsu:Id=”EstablishSessionPolicy”>
      <!—Profile -->
      <nemosec:Profile URI=”&nemo;/profile/main”/>
      <!—Protocol -->
      <nemop:ProtocolAssertion>
         <nemop:Reference URI=”&nemosec;/secure-protocol/secure-
            conversation/1.0”/>
      </nemop:ProtocolAssertion>
   </wsp:Policy>

   <!—Session Request Policy -->
   <wsp:Policy wsu:Id=”SessionRequestPolicy”>

      <!—Protocol -->
      <nemop:ProtocolAssertion>
         <nemop:Reference URI=”&nemosec;/secure-protocol/secure-
            conversation/1.0”/>
         <nemosec:Step type=”establishment-request”>
      </nemop:ProtocolAssertion>

      <!—Service’s encryption key -->
      <wssp:SecurityToken
         nemosec:Usage=”&nemosec;/secure-
            protocol/secure-conversation/1.0#establishment-
            request-encryptionKey”/>

      <!—client’s signature key -->
      <wssp:SecurityToken
            nemosec:Usage=”&nemosec;/secure-protocol/secure-
            conversation/1.0#establishment-request-signingKey”>
         <wsse:TokenType>http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-x509-token-profile-1.0#X509PKIPathv1</wssp:TokenType>
         <wssp:TokenIssuer>
           ...Trusted Roots...
         </wssp:TokenIssuer>
      </wssp:SecurityToken>

      <nemop:Nonce
         nemop:Usage=”&nemosec;/secure-
            protocol/secure-
            conversation/1.0/policyAssertion#nonce”/>
      <wssp:MessageAge Age="3600"
```

```
         nemop:Usage=”&nemosec;/secure-
            protocol/secure-
            conversation/1.0/policyAssertion#timestamp”/>

      <!— The keys to be used for confidentiality and integrity -->
      <!-  are specified by the protocol binding -->
      <wssp:Confidentiality
         nemop:Usage=”&nemosec;/secure-
            protocol/secure-
            conversation/1.0/policyAssertion#confidentiality”>
         <wssp:MessageParts Dialect=”&nemop;#part”>
            wsp:Body()
         </wssp:MessageParts>
      </wssp:Confidentiality>
      <wssp:Integrity
         nemop:Usage=”&nemosec;/secure-
            protocol/secure-
            conversation/1.0/policyAssertion#integrity”>
         <wssp:MessageParts Dialect=”&nemop;#part”>
            wsp:Body()
            nemop:Token(&nemosec;/…/establishment-request-
                  timestamp)
            nemop:Token(&nemosec;/…/establishment-request-toNode)
         </wssp:MessageParts>
      </wssp:Integrity>

      <!-- client’s Role -->
      <wsp:ExactlyOne>
         <wssp:SecurityToken>
            <wssp:TokenType>wsse:SAMLAssertion</wssp:TokenType>
            <wssp:TokenIssuer>...Trusted Roots...
            </wssp:TokenIssuer>
            <wssp:Claims>
              <nemop:NameValuePairDescription
                  Name=”role”
                  Namespace=”&nemop;/attribute”>
                  <nemop:ValuePattern>
                      MyClientRole
                  </nemop:ValuePattern>
              </nemop:NameValuePairDescription>
            </wssp:Claims>
         </wssp:SecurityToken>

         <wssp:SecurityToken>
            <wssp:TokenType>wsse:SAMLAssertion</wssp:TokenType>
            <wssp:TokenIssuer>...Trusted Roots...
            </wssp:TokenIssuer>
            <wssp:Claims>
              <nemop:NameValuePairDescription
                  Name=”role”
                  Namespace=”&nemop;/attribute”>
                  <nemop:ValuePattern>
```

```
                      MyAlternateClientRole
                  </nemop:ValuePattern>
              </nemop:NameValuePairDescription>
            </wssp:Claims>
         </wssp:SecurityToken>
      </wsp:ExactlyOne>
   </wsp:Policy>

   <!–Session Response Policy -->
   <wsp:Policy wsu:Id=”SessionResponsePolicy”>

      <!–Protocol -->
      <nemop:ProtocolAssertion>
         <nemop:Reference URI=”&nemosec;/secure-protocol/secure-
            conversation/1.0”/>
         <nemosec:Step type=”establishment-response”>
      </nemop:ProtocolAssertion>

      <!–client’s encryption key -->
      <wssp:SecurityToken
         nemosec:Usage=”&nemosec;/secure-protocol/secure-
         conversation/1.0#establishment-response-encryptionKey”>
         <wsse:TokenType>http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-x509-token-profile-1.0#X509PKIPathv1</wssp:TokenType>
         <wssp:TokenIssuer>
            ...Trusted Roots...
         </wssp:TokenIssuer>
      </wssp:SecurityToken>

      <!–service’s public signature key -->
      <wssp:SecurityToken
         nemosec:Usage=”&nemosec;/secure-
            protocol/secure-conversation/1.0#establishment-
            response-signingKey”/>

      <nemop:Nonce
         nemop:Usage=”&nemosec;/secure-
            protocol/secure-
            conversation/1.0/policyAssertion#nonce”/>
      <wssp:MessageAge Age="3600"
         nemop:Usage=”&nemosec;/secure-
            protocol/secure-
            conversation/1.0/policyAssertion#timestamp”/>
      <wssp:Confidentiality
         nemop:Usage=”&nemosec;/secure-
            protocol/secure-
            conversation/1.0/policyAssertion#confidentiality”>
         <wssp:MessageParts Dialect=”&nemop;#part”>
            wsp:Body()
         </wssp:MessageParts>
      </wssp:Confidentiality>
      <wssp:Integrity
```

```
         nemop:Usage=”&nemosec;/secure-
            protocol/secure-
            conversation/1.0/policyAssertion#integrity”>
         <wssp:MessageParts Dialect=”&nemop;#part”>
            wsp:Body()
            nemop:Token(&nemosec;/…/establishment-response-
                  timestamp)
            nemop:Token(&nemosec;/…/establishment-response-
                  nonce)
            nemop:Token(&nemosec;/…/establishment-response-
                  toNode)
         </wssp:MessageParts>
      </wssp:Integrity>
   </wsp:Policy>
   ...
</wsdl:definitions>
```

6 NEMO Discovery/Inspection Bindings

6.1 *Overview*

This section specifies XML-related bindings pertaining to NEMO Inspection and Discovery.

- Discovery – the ability to search for services offered by NEMO nodes based on different criteria, and to obtain references to where we can bind to those services for access.
- Inspection – given a reference to a NEMO node, the ability to query it about certain well-defined attributes (metadata) in regards to its state, such as the descriptions of the policy related to the services it publicly offers.

6.2 *Service Discovery Binding*

6.2.1 Overview

Discovery is the ability to search for services offered by NEMO nodes based on different criteria, and to obtain matching references to those services. It embodies two separate aspects of Service Oriented Architectures related to locating and inspecting networked resources:

- Querying managed registries (databases or directory services) to locate resources
- Dynamic, decentralized advertising and locating of (generally transient) resources.

NEMO can support many different types of bindings for discovery and inspection. The following sections describe the currently defined bindings. NEMO nodes may implement and support more then one binding.

6.2.2 WS-Discovery

This binding is based on top of the evolving WS-Discovery specifications [WS-Discovery].

6.2.2.1 WS-Discovery Announcement

6.2.2.1.1 Description

This binding supports a mode of functionality that is intended for small unabridged subnets including Personal Area Networks (PANs) and small unsegmented Local Area Networks (LANs), but not segmented LAN or Wide Area Networks (WANs). The initial version of this binding is intended for use with IP networks, although it can be adapted to other types of networks. This binding is generally implemented on top of multicast protocols and transports such as UDP. It also has a unicast aspect in some phases of the protocols. The following bindings use a profile of WS-Discovery that leverages a UPnP-like type of discovery.

6.2.2.1.2 Requirements (Normative)

NEMO nodes supporting this binding and offering services shall comply with WS-Discovery [WS-Discovery]. In particular, the following message protocols will be supported:

- Probe – A node announcing its desire for a particular type of service by multicasting a (Probe) message in conjunction with a node's listening for such announcements.

- ProbeMatch – A node response with a unicast message (ProbeMatch) directed at the source of the probe if it has matching target services.

- Hello – A node announcing its service capabilities by multicasting a (Hello) message at well-defined events (such as joining a network, time of day, etc.) in conjunction with a node's listening for such announcements.

- Bye – A node announcing that a service capability is no longer available by multicasting a (Bye) message at well-defined events (such as leaving a network) in conjunction with a NEMO node's listening for such announcements.

- Resolve – A node announcing its desire for a particular target service based on transport-neutral service address (ID) multicasting a (Resolve) message in conjunction with a node's listening for such announcements.

- ResolveMatch – A node response with a unicast message (ResolveMatch) directed at the source of the Resolve if it has matching target services.

Multicast messages will be expected on the following communication endpoint: (Port=3702, IPV4=239.255.255.250).

As defined by the specification, Probes may support a wide variety of criteria for matching against target services, but at a minimum nodes will support matching based on service type.

As defined by the specification, ProbeMatches and ResolveMatches may describe the communication endpoint for interacting with a node in many different ways using endpoint references. In addition, a transport-specific communication address may also be conveyed in the context of a `<wsd:XAddrs>` element.

6.2.2.2 WS-Discovery Discovery Proxy

6.2.2.2.1 Description

This binding supports a mode of functionality that is intended for situations where announcement (multicast) type of discovery as described above is inappropriate. This binding is appropriate for all types of network topologies where specific communication endpoints can be addressed.

6.2.2.2.2 Requirements (Normative)

NEMO nodes implementing this binding shall comply with WS-Discovery, Discovery Proxy Service [WS-Discovery]. The actual Discovery Proxy Service interface definition is out of the scope of the current WS-Discovery specification. We have defined a simple one that is consistent with the intention and the behavior of the specification.

This binding is generally implemented on top of unicast protocols and transports such as TCP. The following bindings define a profile of WS-Discovery that leverages a registry-like type of discovery.

In particular, this service provides the ability to query against a registry of services, not just against those belonging to the target node, but also against any services the target node is aware of.

- One round, request/response messaging pattern.

- The input message (request) consists of a standard WS-Discovery Probe in a simple wrapper. It will match against all services that it knows about, not just its own target services.
- The output message (response) consists of a bundle of zero or more WS-Discovery Probe response(s) representing all matching target services known by the node.
- The service is of type DiscoveryProxy, in the http://nemo.intertrust.com/2004/discovery namespace.

6.2.2.2.3 Security

While this binding is amendable for use with the NEMO Basic Secure Messaging Protocol, one areas that needs further work is the issue of signing QNames used within these specifications.

6.2.2.3 Matching Criteria

6.2.2.3.1 Matching By Service Type

WS-Discovery supports matching based on service address, service type, and extensible matching predicates called *scopes*, which are evaluated in the context of a given service.

We define type-based matching based on web service port types. A service's types are defined by one or more port types defined by a QName in a defined namespace.

6.2.2.3.2 Extended Forms of Matching (By Scope)

It is possible to define new types of criteria for matching with WS-Discovery. This general facility is called *scope*s.

We currently have defined some new optional types of scopes that allow for the following types of matching:

1. Matching by NEMO node information, such as node ID:

```
<wsd:Scope
MatchBy="http://nemo.intertrust.com/discovery/scope/matchbynodeinfo">
   <xsd:element ref="nemoc:NodeInfo">
</wsd:Scope>
```

If this scope is supported, and if the specified node information matches for a given node, then the scope is true, else the scope is false.

2. Matching by support roles. The role is represented in terms of a namespace, a role value, and an optional issuer.

```
<wsd:Scope
MatchBy="http://nemo.intertrust.com/discovery/scope/matchbyrole">
   <xsd:element ref="nemodisc:RoleScopeCriteria">
</wsd:Scope>
```

3. Matching based on security policy-related tokens, such as trust anchors.

```
<wsd:Scope
MatchBy="http://nemo.intertrust.com/discovery/scope/matchbyrole">
   <xsd:element ref="nemodisc:PolicyTokenScopeCriteria ">
    </wsd:Scope>
```

If this scope is supported, and if the specified policy token information matches for a given node, then the scope is true, else the scope is false.

Please see §6.4 and §6.5 for schemas and examples related to the forms of matches possible with this binding.

6.2.2.3.3 Rules for Matching

In this binding, the basic rules for matching for discovery are pretty simple. A discovery query may consist of any of the following:

1. Query based on service address.
2. Query based on one or more types, where type matches are logically OR-ed .
3. Query based on one or more scopes, where scopes must be of the **same** dialect, and scope matches are logically OR-ed .
4. Query 2 and Query 3 in combination logically AND-ed.

6.2.2.4 Service References, Properties and Parameters

If a matching service is found, and a response returned, or if a service's availability is announced, the message will contain a description of the service. This either allows the service to be directly interacted with or it provides enough information to bootstrap another process, such as inspection, via the service. We call this description a service reference. The types of service references defined in this binding are web service references.

In this profile, the web service reference is embodied in terms of a WS-Addressing endpoint reference, which will contain at least the following:

- Service Name
- Service Port Type(s)
- Service Address (transport neutral)
- Service Transport Address (Port Address)

This binding also defines an optional reference parameter that may be returned to indicate the location of a supporting service that can be used to perform inspection in regards to the matched service in order to obtain metadata. An example of needing this parameter would be discovering a service and needing to obtain its WSDL:

```
<nemoc:InspectionReference
xmlns:nemoc=http://nemo.intertrust.com/2004/core>
   <wsa:EndpointReference>
      <wsa:Address>
      http://localhost:9084/SimpleDiscovery/services/MetadataExchange
      </wsa:Address>
   </wsa:EndpointReference>
</nemoc:InspectionReference>
```

6.2.3 SSDP-based Discovery

This binding is based on the Simple Service Discovery Protocol (SSDP) 1.0 specification [SSDP] for discovery, and specifically relates to how NEMO nodes and their related services may be discovered within the context of a local area network using the Simple Service Discovery Protocol.

6.2.3.1 Goals

SSDP has flourished in the context of standards such as UPnP as a way of exposing and interacting with devices and their services, particularly in the context of home environments because of its simplicity.   It is important that we leverage existing and emerging standards in this area that balance the introduction of new capabilities with retaining compatibility with existing UPnP environments. The goals of this discovery binding are to allow NEMO services to be discovered via SSDP and then later inspected for more detailed information about the services.

6.2.3.2 Discovery Protocol Support

NEMO nodes supporting this binding will comply with the protocols for supporting service discovery and description, as set forth in the Simple Service Discovery Protocol 1.0 [SSDP] specifications.

While SSDP is used as the discovery protocol in UPnP, and while some deployments may choose to use this binding in a way that fully interoperates with UPnP deployments, there is no obligation to do so.

The current binding supports a node matching against its services based on one type of specified search criteria conveyed in the **ST** header associated with an **ssdp:discover** request. The form of the data conveyed in the **ST** header is a URI. NEMO is **neutral** as to the exact syntax and semantics of the URI. Other profiles should build on top of this specification in order to create matching schemes that mandate particular usage patterns. In supporting this binding, however, NEMO does mandate that at a **minimum** matching based on service type, where matching is done by exact comparison of the URI, is supported.

With NEMO, it is possible to convey a variety of different search criteria in this header. Here are some **examples** of how you could express different criteria:

1. Search by Service Type.

   This mode of searching supports searching by service types. The currently supported bindings for NEMO use WSDL port types to denote unique service types. The form of the **ST** header in this case is:

   <service namespace>#<service name>

   Example:

   http://nemo.intertrust.com/services#OctopusLicenseService

2. Search by NEMO Attribute.

This mode of searching supports searching by a designated attribute, such as a role in NEMO. The service must support the designated role. The form of the **ST** header in this case is:

<attribute namespace>#<attribute name>#<attribute value>

Example:

http://nemo.intertrust.com/roles#role#LicenseService

3. Search by Trust Anchor

This mode of searching supports searching by a designated trust anchor in NEMO. The service's authentication trust anchor must be the designated trust anchor. The form of the **ST** header in this case is:

<trust anchor distinguished name>

Example:

urn:trustanchor:1.3.6.1.4.1.7584.1.1.1=urn:ca:SystemD

6.2.3.3 Service Identifier

Per the SSDP specification, there must be a way to associate an identifier with a service instance (USN). While NEMO does not mandate the exact form of this identifier, implementers of this binding will support returning an appropriate USN. The USN may be used in subsequent interactions with the inspection service to identify the service instance.

Example:

USN: http://localhost:9082/MarlinSettop/services/OctopusLicense

6.2.3.4 Service Inspection

In order to actually communicate with a service, it may be necessary for you to go through a service inspection interaction, for example if the USN for the discovered service is insufficient to provide enough information to resolve and use the service. NEMO defines a binding based on WS-MetadataExchange for inspection, where you can obtain additional information in regards to using the service based on a unique service ID. If a node finds an appropriate match, it will send back an SSDP discovery response, where the **Location** header contains a service endpoint reference in the form of a URI for the inspection service to be used.

Example:

Location: http://localhost:8080/service/inspection

After inspecting a service to obtain its location and other information necessary for interacting with it, a node may cache the result to avoid going through subsequent inspection interactions.

6.2.3.5 Inspection Policy Identifier

The inspection service, like any other NEMO service, may be securely policy managed, and a client must interact with the service with a well-defined NEMO security protocol.

We introduce a new HTTP header (**Inspection-Policy-Id**) used in SSDP discovery responses and in presence announcements that will designated the security policy associated with using the inspection service. This policy may be that no security policy is in effect. The form of the policy identifier is a URI.

Example:

Inspection-Policy-Id: urn:marlinpolicy:0001

6.2.4 Presence Announcements

In addition to NEMO nodes supporting discovery **ssdp:discover** messages, NEMO nodes may optionally support the SSDP **ssdp:alive** and **ssdp:bye** messages for alerting interested parties to the status of their services.

6.2.5 Additional UPnP-Related Extensions

In addition to supporting SSDP-based discovery, NEMO nodes may optionally support additional extensions that make interacting with a UPnP environment more seamless. Nodes supporting this binding will comply with the protocols for supporting addressing, discovery, and description, as set forth in the UPnP Device Architecture Version 1.0 [UPnP] specifications.

These proposed extensions allow a NEMO nodes to be exposed as a UPnP device. The NEMO node may be represented as a virtual device or may correspond to an actual physical device. In particular, these extensions:

1. Allow NEMO nodes to expose services that fit the traditional UPnP usage model as UPnP services. A NEMO node may choose to expose and describe services that are completely compliant with traditional UPnP usage models, or not.

2. Allow NEMO nodes to expose information about the node and about services that are beyond the current scope of UPnP usage models. For example, certain services may require authentication and authorization capabilities that aren’t directly supported by UPnP protocols.

3. Allow non-NEMO node UPnP devices to discover NEMO nodes and their services. A standard UPnP device should have the capability to discover NEMO nodes and their extended properties. A UPnP device may be unaware of or choose not to process any properties and services that don’t fit the standard usage model, but the capability exists for discovery.

4. Allow NEMO nodes to discover, inspect, and invoke services on non-NEMO node-ed UPnP devices. In this case, the NEMO node will act in a capacity that supports the UPnP control point.

In this binding we deal with the following cases:

1. One-to-one correspondence between NEMO node and actual device, i.e., a device fully encapsulates a NEMO node. In this case, the `upnp:rootdevice` type and associated device UUID will be bound to the device itself.

2. NEMO node without corresponding device. In this case, the NEMO node will represent itself as a UPnP root device with appropriate device type, UUID and other necessary information to describe it.

3. NEMO nodes may act as UPnP control points for the purposes of actually performing searches or receiving events.

In terms of protocol support more specifically:

1. NEMO nodes will support UPnP discovery advertisement, and when a node is added to a network, the UPnP discovery protocol will allow that node to advertise its services to UPnP control points.
2. NEMO nodes will support discovery search requests for M-Search and respond accordingly.
3. NEMO nodes may expose a UPnP device description that contains a set of extensions for supporting properties and services that are beyond standard UPnP usage models.

6.2.5.1 Extensions to Device Description

As previously mentioned, a NEMO node may be inspected after being discovered using SSDP. The information related to inspection of the node is in the form of an XML encoded device description, obtained via HTTP, whose location is communicated back in a discovery response. In addition to the standard UPnP information contained in the description, a NEMO node may include an extension that defines additional information.

The extension type X_NEMONode-Extension in the http://nemo.intertrust.com/discovery/upnp namespace defines properties specifically related to NEMO nodes and services that fall outside the scope of the traditional UPnP usage model.

In particular, the extension may include:

1. NEMO node info, which can include node ID and security credentials related to the node, such as SAML roles defined for the role and public keys associated with usage of services.
2. List of services defined for the node, including an endpoint reference for contacting the service. We use WS-Addressing Endpoint references to describe the service endpoints.
3. A description of how to further inspect the services that are specified in the extension. Here we leverage the existing mechanism we defined for inspection with WS-Discovery, providing an inspection reference (`nemoc:InspectionReference`), which could contain a pointer to where the WSDL could be obtained or a pointer to a service that could be used for further inspection, such as inspection services based on WS-MetadataExchange.

Section 6.4.3.1 describes the schemas for the device description extension and §6.5.7 provides an example descriptor.

6.2.5.2 Extensions to Service Description

A NEMO node is required to expose one or more services to be UPnP compliant. The descriptions of these services will be exposed to other UPnP entities in the form of SCP descriptors.

Some form of the SCP descriptor may vary depending on the intended usage. The following are three potential forms:

1. Standard SCPD with both UPnP service actions and events. This is the form of descriptor that today is commonly used with UPnP entities.

2. SCPD with no actions or events and with a NEMO service extension. This is the form of descriptor where it is not possible to describe the NEMO service input and output messages based on the standard SCP description language. The form supports discovery of the service over UPnP, but further understanding of the service requirements will either have to be made through local means or via processing of the NEMO service extension.

3. SCPD with actions and events and with a NEMO service extension. This is the form of the service description where we support the traditional UPnP usage model but include additional service information in the extension that could be used by clients that are NEMO-aware, but it is not mandatory across all clients.

The extension type X_NEMOService-Extension in the http://nemo.intertrust.com/discovery/upnp namespace defines properties specifically related to a NEMO node's service that fall outside the scope of the traditional UPnP usage model.

In particular, the extension may include:

Reference to a WSDL description or the actual WSDL itself for a service if it cannot be adequately described in SCP.

A description of how to further inspect the service that is being specified. Here we leverage the existing mechanism we defined for inspection with WS-Discovery, providing an inspection reference (nemoc:InspectionReference).

Section 6.4.3.2 describes the schemas for the service description extension, and §6.5.8 provides an example descriptor.

6.2.5.3 UPnP Service Security

NEMO services are secured based on extensible security protocols in the web services community.

Even though most standard UPnP environments either lack general mechanisms for service level security or secure the contents of a service message at the business level payload, there do exist proposals for standard UPnP based service security, including the specifications based on Device Security and Security Console V 1.0 (http://www.upnp.org/standardizeddcps/security.asp).

6.3 Inspection Binding

6.3.1 Description

Given a reference to a NEMO node, inspection provides the ability to query it about certain well-defined attributes (metadata), such as the descriptions of the policy related to the services it publicly offers or WSDL or other more general NEMO node information.

In this current binding we are currently addressing inspection of metadata surrounding target service endpoints exported by NEMO nodes. Inspection is supported as a well-defined service a node exports.

6.3.2 Requirements (Normative)

NEMO nodes supporting this binding and offering services shall comply with the Web Services Metadata Exchange [WS-MetadataExchange] specification. As stated in the specification, the following message protocols will be supported:

- MetadataGet: a node initiates a Metadata GET request, specifying a target service endpoint and potentially a dialect and/or identifier which designate the specific type of metadata or specific metadata object to retrieve. In response, the requested metadata for the target service endpoint is returned in terms of its actual content and/or a reference to where to obtain the content via a GET operation. If no dialect or identifier is provided, all metadata for the target service endpoint is returned.
- Get: a node initiates a GET Request specifying a reference to a specific instance of metadata. In response, the actual metadata is returned.

In addition to the standard types of metadata dialects for WSDL and Policy, we define a new dialect for retrieving NEMO node information related to a given service address:

http://nemo.intertrust.com/2004/inspection/mex/nemonodeinfo

At a minimum, NEMO nodes **may** understand and support retrieving metadata formats based on the following standard dialects:

- XML Schema Version 1.0
- WSDL 1.1
- WS-Policy expression

Additional dialects may also be defined, such as:

- NEMO Node Information

6.4 *Schema and Abstract Web Service Definitions*

6.4.1 NEMO Node Description

```
<xsd:schema targetNamespace="http://www.intertrust.com/core"
nemoc="http://www.intertrust.com/core" xmlns:wsse="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
elementFormDefault="qualified" attributeFormDefault="qualified"
version="0.5">
<xsd:complexType name="NodeInfo">
<xsd:complexContent>
<xsd:extension base="nemoc:Base">
<xsd:sequence>
<xsd:element name="NodeId" type="xsd:uri" minOccurs="0"/>
<xsd:element ref="wsse:SecurityTokenReference" minOccurs="0"
maxOccurs="unbounded">
</xsd:sequence>
</xsd:extension>
</xsd:complexContent>
```

```
</xsd:complexType>
<xsd:element name="NodeInfo" type="nemoc:NodeInfo"/>
</xsd:schema>
```

6.4.2 Discovery

6.4.2.1 WS-Discovery Scope-Related Definitions

```
<xsd:schema targetNamespace="http://nemo.intertrust.com/discovery/wsd"
nemodisc="http://nemo.intertrust.com/discovery/wsd"
nemoc="http://www.intertrust.com/core"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
elementFormDefault="qualified" attributeFormDefault="qualified"
version="0.5">

<xsd:complexType name="RoleAttributeDesignator">
<xsd:complexContent>
   <xsd:extension base="nemoc:Base">
      <xsd:sequence>
         <xsd:element name="issuer" type="xsd:string" minOccurs="0">
         <xsd:element name="name" type="xsd:string">
         <xsd:element name="value" type="xsd:string">
      </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
<xsd:element name="RoleAttributeDesignator"
type="nemodisc:RoleAttributeDesignator"/>

<xsd:complexType name="RoleScopeCriteria">
<xsd:complexContent>
   <xsd:extension base="nemoc:Base">
      <xsd:sequence>
         <xsd:element name="targettype" type="xsd:string">  <!-- NODE,
SERVICE -->
         <xsd:element ref="nemodisc:RoleAttributeDesignator"
minOccurs="1" maxOccurs="unbounded">
      </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
<xsd:element name="RoleScopeCriteria"
type="nemodisc:RoleScopeCriteria"/>


<xsd:complexType name="PolicyTokenScopeCriteria">
<xsd:complexContent>
   <xsd:extension base="nemoc:Base">
      <xsd:sequence>
         <xsd:element ref="wsse:SecurityTokenReference">
      </xsd:sequence>
   </xsd:extension>
</xsd:complexContent>
</xsd:complexType>
<xsd:element name="PolicyTokenScopeCriteria"
type="nemodisc:PolicyTokenScopeCriteria"/>

</xsd:schema>
```

6.4.2.2 WS-Discovery Discovery Proxy Schema Definitions and WSDL

```
<wsdl:definitions
   xmlns="http://schemas.xmlsoap.org/wsdl/"
      targetNamespace='http://nemo.intertrust.com/2004/discovery'
      xmlns:tns="http://nemo.intertrust.com/2004/discovery"
   xmlns:ds='http://schemas.xmlsoap.org/ws/2004/02/discovery'
      xmlns:wsdl='http://schemas.xmlsoap.org/wsdl/'
      xmlns:wsdlsoap='http://schemas.xmlsoap.org/wsdl/soap/'
      xmlns:xs='http://www.w3.org/2001/XMLSchema'
name="DiscoveryProxyService">

  <wsdl:types>
    <xs:schema xmlns="http://nemo.intertrust.com/2004/discovery"
targetNamespace="http://nemo.intertrust.com/2004/discovery">
         <xs:import
namespace='http://schemas.xmlsoap.org/ws/2004/02/discovery'
location='discovery.xsd' />
   <xs:element name="ProxyProbe">
            <xs:complexType>
              <xs:sequence>
            <xs:element ref="ds:Probe"/>
            </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:element name="ProxyProbeMatch">
            <xs:complexType>
              <xs:sequence>
            <xs:element ref="ds:ProbeMatches minOccurs="0"/>
            </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:element name="ProxyResolveMatch">
            <xs:complexType>
              <xs:sequence>
            <xs:element ref="ds:ResolveMatch" minOccurs="0"/>
            </xs:sequence>
      </xs:complexType>
   </xs:element>
    </xs:schema>
  </wsdl:types>

  <wsdl:message name="ProxyProbeMsg">
    <wsdl:part name='body' element="tns:ProxyProbe" />
  </wsdl:message>

 <wsdl:message name="ProxyProbeMatchMsg">
    <wsdl:part name='body' element="tns:ProxyProbeMatch" />
  </wsdl:message>

  <wsdl:message name="ProxyResolveMsg">
  </wsdl:message>

 <wsdl:message name="ProxyResolveMatchMsg">
    <wsdl:part name='body' element="tns:ProxyResolveMatch" />
  </wsdl:message>


  <wsdl:portType name="DiscoveryProxy">
    <wsdl:operation name='ProxyProbeOp'>
      <wsdl:input message="tns:ProxyProbeMsg"/>
   <wsdl:output message="tns:ProxyProbeMatchMsg"/>
    </wsdl:operation>
    <wsdl:operation name='ProxyResolveOp'>
      <wsdl:input message="tns:ProxyResolveMsg"/>
   <wsdl:output message="tns:ProxyResolveMatchMsg"/>
    </wsdl:operation>
  </wsdl:portType>

</wsdl:definitions>
```

6.4.3 Inspection

6.4.3.1 NEMO UPnP Device Description Extension

```
<xsd:schema targetNamespace=http://nemo.intertrust.com/discovery/upnp
nemoupnp="http://nemo.intertrust.com/discovery/upnp"
nemoc="http://www.intertrust.com/core"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
elementFormDefault="qualified" attributeFormDefault="qualified"
version="0.5">

<xsd:complexType name="ServiceList">
   <xsd:complexContent>
      <xsd:extension base="nemoc:Base">
      <xsd:sequence>
      <xsd:element ref="nemoc:InspectionReference" minOccurs="0">
      <xsd:element ref="wsa:EndpointReference"
minOccurs="0" maxOccurs="unbounded">
            </xsd:sequence>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>
   <xsd:element name="ServiceList" type="nemoupnp:ServiceList"/>

   <xsd:complexType name="X_NEMONode-Extension">
      <xsd:complexContent>
         <xsd:extension base="nemoc:Base">
            <xsd:sequence>
              <xsd:element ref="nemoc:NodeInfo">
              <xsd:element ref="nemoupnp:ServiceList" minOccurs="0">
            </xsd:sequence>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>
   <xsd:element name="X_NEMONode-Extension" type="nemoupnp:X_NEMONode-
Extension"/>
</xsd:schema>
```

6.4.3.2 NEMO UPnP Service Description Extension

```
<xsd:schema targetNamespace=http://nemo.intertrust.com/discovery/upnp
nemoupnp="http://nemo.intertrust.com/discovery/upnp"
nemoc="http://www.intertrust.com/core"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl"
elementFormDefault="qualified" attributeFormDefault="qualified"
version="0.5">

   <xsd:complexType name="X_NEMOService-Extension">
      <xsd:complexContent>
         <xsd:extension base="nemoc:Base">
            <xsd:sequence>
              <xsd:element ref="wsdl:definitions" minOccurs="0">
              <xsd:element ref="nemoc:InspectionReference"
minOccurs="0">
            </xsd:sequence>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>
   <xsd:element name="X_NEMOService-Extension"
type="nemoupnp:X_NEMOService-Extension"/>
</xsd:schema>
```

6.5 *Sample Messages*

6.5.1 WS-Discovery Probe for a Service that Supports DiscoveryProxy.

```
<soap:Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:nemo="http://nemo.intertrust.com/2004/core"
```

```
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
xmlns:wsd="http://schemas.xmlsoap.org/ws/2004/10/discovery"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

<soap:Header>
   <wsa:Action>
   http://schemas.xmlsoap.org/ws/2004/10/discovery/Probe
   </wsa:Action>
   <wsa:MessageID>uuid:1097696212363</wsa:MessageID>
   <wsa:To>urn:schemas-xmlsoap-org:ws:2004:10:discovery</wsa:To>
   <wsa:ReplyTo>
   <wsa:Address>
      http://localhost:9080/SimpleDiscovery/services/Discovery
     </wsa:Address>
     </wsa:ReplyTo>
</soap:Header>

<soap:Body>
<wsd:Probe>
<wsd:Types xmlns:dtypens0="http://nemo.intertrust.com/2004/discovery">
dtypens0:DiscoveryProxy
</wsd:Types>
</wsd:Probe>
</soap:Body>
</soap:Envelope>
```

6.5.2 WS-Discovery ProbeMatch for a Service that Supports DiscoveryProxy.

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:nemo="http://nemo.intertrust.com/2004/core"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
xmlns:wsd="http://schemas.xmlsoap.org/ws/2004/10/discovery"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

<soap:Header>
   <wsa:Action>
      http://schemas.xmlsoap.org/ws/2004/10/discovery/ProbeMatch
   </wsa:Action>
   <wsa:MessageID>uuid:1097696212463</wsa:MessageID>
   <wsa:RelatesTo>uuid:1097696212363</wsa:RelatesTo>
   <wsa:To>http://localhost:9080/SimpleDiscovery/services/Discovery
   </wsa:To>
</soap:Header>
```

```

<soap:Body>
<wsd:ProbeMatches>
   <wsd:ProbeMatch>
      <wsa:EndpointReference>
         <wsa:Address>
         http://localhost:9084/SimpleDiscovery/services/DiscoveryProxy
         </wsa:Address>
         <wsa:ReferenceParameters>
            <nemo:InspectionReference
            xmlns:nemo=http://nemo.intertrust.com/2004/core>
              <wsa:EndpointReference>
                  <wsa:Address>
                  http://localhost:9084/SimpleDiscovery/services/
MetadataExchange
                  </wsa:Address>
              </wsa:EndpointReference>
            </nemo:InspectionReference>
         </wsa:ReferenceParameters>
         <wsa:ReferenceProperties>
            <nemo:NodeInfo
xmlns:nemo="http://nemo.intertrust.com/2004/core">
              <nemo:NodeId
xmlns:nemo="http://nemo.intertrust.com/2004/core">
urn:node002
              </nemo:NodeId>
            </nemo:NodeInfo>
         </wsa:ReferenceProperties>
         <wsa:PortType
xmlns:svcns0="http://nemo.intertrust.com/2004/discovery">
svcns0:DiscoveryProxy
         </wsa:PortType>
         <wsa:ServiceName
xmlns:svcns0="http://nemo.intertrust.com/2004/discovery">
svcns0:DiscoveryProxyService
         </wsa:ServiceName>
      </wsa:EndpointReference>
      <wsd:Types
xmlns:dtypens0="http://nemo.intertrust.com/2004/discovery">
dtypens0:DiscoveryProxy
      </wsd:Types>
      <wsd:Scope
MatchBy="http://nemo.intertrust.com/discovery/scope/matchbynodeinfo">
         <nemo:NodeInfo
xmlns:nemo="http://nemo.intertrust.com/2004/core">
            <nemo:NodeId
xmlns:nemo="http://nemo.intertrust.com/2004/core">
urn:node002
            </nemo:NodeId>
         </nemo:NodeInfo>
      </wsd:Scope>
      <wsd:XAddrs>
```

```
http://localhost:9084/SimpleDiscovery/services/DiscoveryProxy
      </wsd:XAddrs>
      <wsd:MetadataVersion>1</wsd:MetadataVersion>
   </wsd:ProbeMatch>
</wsd:ProbeMatches>
</soap:Body>
</soap:Envelope>
```

6.5.3 WS-Discovery Hello Announcement to Announce Availability of Service

```
<soap:Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:nemo="http://nemo.intertrust.com/2004/core"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
xmlns:wsd="http://schemas.xmlsoap.org/ws/2004/10/discovery"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

<soap:Header>
   <wsa:Action>
http://schemas.xmlsoap.org/ws/2004/10/discovery/Hello
   </wsa:Action>
   <wsa:MessageID>uuid:1097695419415</wsa:MessageID>
   <wsa:To>urn:schemas-xmlsoap-org:ws:2004:10:discovery</wsa:To>
   <wsd:AppSequence InstanceId="00032111111" MessageNumber="1"/>
</soap:Header>

<soap:Body>
<wsd:Hello>
   <wsa:EndpointReference>
      <wsa:Address>
http://localhost:9080/SimpleDiscovery/services/TestE
      </wsa:Address>
      <wsa:ReferenceParameters>
         <nemo:InspectionReference
xmlns:nemo="http://nemo.intertrust.com/2004/core">
            <wsa:EndpointReference>
              <wsa:Address>
http://localhost:9080/SimpleDiscovery/services/MetadataExchange
              </wsa:Address>
            </wsa:EndpointReference>
         </nemo:InspectionReference>
      </wsa:ReferenceParameters>
      <wsa:ReferenceProperties>
         <nemo:NodeInfo
xmlns:nemo="http://nemo.intertrust.com/2004/core">
            <nemo:NodeId
xmlns:nemo="http://nemo.intertrust.com/2004/core">
urn:node000
            </nemo:NodeId>
         </nemo:NodeInfo>
      </wsa:ReferenceProperties>
      <wsa:PortType xmlns:svcns0="http://www.intertrust.com/services">
svcns0:TestE</wsa:PortType>
      <wsa:ServiceName
xmlns:svcns0="http://www.intertrust.com/services">
svcns0:TestEService
      </wsa:ServiceName>
   </wsa:EndpointReference>
   <wsd:Types xmlns:dtypens0="http://www.intertrust.com/services">
dtypens0:TestE
   </wsd:Types>
   <wsd:Scope
MatchBy="http://nemo.intertrust.com/discovery/scope/matchbynodeinfo">
      <nemo:NodeInfo xmlns:nemo="http://nemo.intertrust.com/2004/core">
         <nemo:NodeId xmlns:nemo="http://nemo.intertrust.com/2004/core">
urn:node000
         </nemo:NodeId>
      </nemo:NodeInfo>
   </wsd:Scope>
   <wsd:XAddrs>
http://localhost:9080/SimpleDiscovery/services/TestE
   </wsd:XAddrs>
   <wsd:MetadataVersion>1</wsd:MetadataVersion>
</wsd:Hello>
</soap:Body>
</soap:Envelope>
```

6.5.4 WS-Discovery Bye to Announce Service Is No Longer Available

```
<soap:Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:nemo="http://nemo.intertrust.com/2004/core"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
xmlns:wsd="http://schemas.xmlsoap.org/ws/2004/10/discovery"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

<soap:Header>
   <wsa:Action>http://schemas.xmlsoap.org/ws/2004/10/discovery/Bye</wsa:
Action>
   <wsa:MessageID>uuid:1097695958833</wsa:MessageID>
<wsa:To>urn:schemas-xmlsoap-org:ws:2004:10:discovery</wsa:To>
   <wsd:AppSequence InstanceId="00032111111" MessageNumber="1"/>
</soap:Header>

<soap:Body>
<wsd:Bye>
   <wsa:EndpointReference>
      <wsa:Address>
```

```
http://localhost:9084/SimpleDiscovery/services/TestA</wsa:Address>
      <wsa:ReferenceParameters>
         <nemo:InspectionReference
xmlns:nemo="http://nemo.intertrust.com/2004/core">
            <wsa:EndpointReference>
             <wsa:Address>
http://localhost:9084/SimpleDiscovery/services/MetadataExchange
             </wsa:Address>
            </wsa:EndpointReference>
         </nemo:InspectionReference>
      </wsa:ReferenceParameters>
      <wsa:ReferenceProperties>
         <nemo:NodeInfo
xmlns:nemo="http://nemo.intertrust.com/2004/core">
            <nemo:NodeId
xmlns:nemo="http://nemo.intertrust.com/2004/core">
urn:node002
            </nemo:NodeId>
         </nemo:NodeInfo>
      </wsa:ReferenceProperties>
   </wsa:EndpointReference>
</wsd:Bye>
</soap:Body>
</soap:Envelope>
```

6.5.5 WS-MetadataExchange Request to Obtain a Service WSDL

```
<soap:Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
xmlns:wsx="http://schemas.xmlsoap.org/ws/2004/09/mex"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

<soap:Header>
   <wsa:Action>
http://schemas.xmlsoap.org/ws/2004/09/mex/GetMetadata/Request
   </wsa:Action>
   <wsa:MessageID>uuid:1097698812758</wsa:MessageID>
   <wsa:To>http://localhost:9082/SimpleDiscovery/services/TestB</wsa:To>
</soap:Header>

<soap:Body>
<wsx:GetMetadata>
   <wsx:Dialect>http://schemas.xmlsoap.org/wsdl/</wsx:Dialect>
   </wsx:GetMetadata>
</soap:Body>
</soap:Envelope>
```

6.5.6 WS-MetadataExchange Response for Obtained Service WSDL

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
xmlns:wsx="http://schemas.xmlsoap.org/ws/2004/09/mex"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

<soap:Header>
   <wsa:Action>
http://schemas.xmlsoap.org/ws/2004/09/mex/GetMetadata/Response
   </wsa:Action>
   <wsa:MessageID>uuid:1097698813489</wsa:MessageID>
   <wsa:RelatesTo>uuid:1097698812758</wsa:RelatesTo>
</soap:Header>

<soap:Body>
<wsx:Metadata>
<wsx:MetadataSectionDialect="http://schemas.xmlsoap.org/wsdl/">
<wsdl:definitions name="TestBService"
targetNamespace="http://www.intertrust.com/services"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:apachesoap="http://xml.apache.org/xml-soap"
xmlns:impl="http://www.intertrust.com/services"
xmlns:intf="http://www.intertrust.com/services"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tnstype="http://www.intertrust.com/services"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">


   <!-- ===================================================
   You can define additional types using the <wsdl:types>
   element, as shown below.
   =================================================== -->
   <wsdl:types>
   </wsdl:types>


   <!-- message declarations -->

   <wsdl:message name="anydata">
      <wsdl:part element="xsd:string" name="body"/>
   </wsdl:message>

   <!-- port type declarations -->
   <wsdl:portType name="TestB">
```

```
      <wsdl:operation name="invoke">
         <wsdl:input message="impl:anydata"/>
         <wsdl:output message="impl:anydata"/>
      </wsdl:operation>
   </wsdl:portType>

   <!-- binding declarations -->
   <wsdl:binding name="TestBSoapBinding" type="impl:TestB">
      <wsdlsoap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="invoke">
         <wsdlsoap:operation
soapAction="http://www.intertrust.com/services/TestB/invoke"/>
         <wsdl:input name="invokeRequest">
            <wsdlsoap:body use="literal"/>
         </wsdl:input>
         <wsdl:output name="invokeResponse">
            <wsdlsoap:body use="literal"/>
         </wsdl:output>

      </wsdl:operation>
   </wsdl:binding>

   <!-- service declarations -->
   <wsdl:service name="TestBService">
      <wsdl:port binding="impl:TestBSoapBinding" name="TestB">
         <wsdlsoap:address
location="http://localhost:9082/SimpleDiscovery/services/TestB"/>
      </wsdl:port>
   </wsdl:service>

</wsdl:definitions>
</wsx:MetadataSection>
</wsx:Metadata>
</soap:Body>
</soap:Envelope>
```

6.5.7 Example UPnP Device Description with NEMO Extension

```
<root xmlns="urn:schemas-upnp-org:device-1-0">
   <specVersion>
      <major>1</major>
      <minor>0</minor>
   </specVersion>
   <device>
      <deviceType>urn:schemas-upnp-org:device:simple:1</deviceType>
      <friendlyName>Generic Simple Device</friendlyName>
      <manufacturer>Intertrust</manufacturer>
      <manufacturerURL>http://www.intertrust.com</manufacturerURL>
      <modelDescription>Generic Simple Device</modelDescription>
      <modelName>Simple</modelName>
      <modelNumber>1.0</modelNumber>
      <modelURL>http://www.intertrust.com</modelURL>
      <serialNumber>1234567890</serialNumber>
      <UDN>uuid:simpledevice</UDN>
      <UPC>123456789012</UPC>

<serviceList>
<service>
   <serviceType>urn:intertrust:service:NEMO-
LicenseCreateor:1</serviceType>
   <serviceId>urn:intertrust:serviceId:LicenseCreator</serviceId>
   <SCPDURL>/LicenseCreator.xml</SCPDURL>
   <ControlURL>/LicenseCreator.xml</ControlURL>
   <eventSubURL></eventSubURL>
</service>
</serviceList>

<X_NEMONode-Extension xmlns="http://nemo.intertrust.com/discovery/upnp">

<nemoc:NodeInfo>
   <nemoc:NodeId>urn:nemo:LicenseCreator</nemoc:NodeId>
   <!-- public encryption Key associated with all node services -->
   <wsse:SecurityTokenReference
nemosec:Usage="http://nemo.intertrust.com/2004/security/secure-
protocol/basic/1.0#request-encryptionKey">
      <wsse:Embedded>
         <wsse:BinarySecurityToken
         ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-x509-token-profile-1.0#X509PKIPathv1"
EncodingType="wsse:Base64Binary">
MIICFzCCAhMwggF8oAMCAQICBEGRH+cwDQYJKoZIhvcNAQEFBQAwHzEdMBsGCisG
AQQBuyABAQETDXVybjpuZW1vOlJvb3QwHhcNMDQxMTA5MTk1MjA3WhcNMDUxMTA5
MTk1MjA3WjApMScwJQYKKwYBBAG7IAEBARMXdXJuOm5lbW86TGljZW5zZUNyZWF0
b3IwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAKS9veyG4L5b4pY9+uRdvOaS
HsURSrYfGsGPo8yLHb2hBVj2R0AlB4N6+ZZbuqJTgh7Y1mIiMjRM/zgMkIpirm9m
8FpiPKRcCm5kA8f5tw3LXbx1LBCp2z5v36q/mSDFLtbEpH0Jd5QRlGRgsME+46DR
LYygVEzSak8s1Muw0tHNAgMBAAGjUjBQMB8GA1UdIwQYMBaAFEFHN1ot9Wtw8fLy
GuCZobAF006CMB0GA1UdDgQWBBSbaoB30xjx80f1qzxAHaS948XjbzAOBgNVHQ8B
Af8EBAMCBSAwDQYJKoZIhvcNAQEFBQADgYEAj0UtfTx7bmCXISvwIcswwVfK8R46
Foh6sbn2khEp1qQkWA6BggRL0tjQQ7DICkcEHSOF+eZrNC+8zUYRGR81Lnyeaqj8
WXMNjfUxIZMqCiTHyrDP6ZbCsUcCAboo/emSyf5wGzLjo6BnGeP+cfMTJ/7iSxY6
TZG2IkVWrfULbVM=
         </wsse:BinarySecurityToken>
      </wsse:Embedded>
   </wsse:SecurityTokenReference>

   <ServiceList>

      <nemoc:InspectionReference
xmlns:nemoc="http://nemo.intertrust.com/2004/core">
```

```
         <wsa:EndpointReference>

   <wsa:Address>http://localhost:9080/services/MetadataExchange</wsa:Address>
         </wsa:EndpointReference>
      </nemoc:InspectionReference>

      <wsa:EndpointReference>
   <wsa:Address>
http://localhost:9080/services/LicenseCreator</wsa:Address>
         <wsa:PortType
xmlns:svcns0="http://www.intertrust.com/services">
svcns0:LicenseCreator</wsa:PortType>
         <wsa:ServiceName
xmlns:svcns0="http://www.intertrust.com/services">
svcns0:LicenseCreator</wsa:ServiceName>
      </wsa:EndpointReference>
   </ServiceList>
</nemoc:NodeInfo>
</X_NEMONode-Extension>
   </device>
</root>
```

6.5.8 Example UPnP Service Description with NEMO Extension

```
<scpd xmlns:"urn:intertrust:service-1-0">
   <specVersion>
      <major>1</major>
      <minor>0</minor>
   </specVersion>
   <serviceStateTable>
      <stateVariable sendEvents="no">
            <name>X_NEMO_Node_ID</name>
            <datatype>uuid</datatype>
      </stateVariable>
   </serviceStateTable>
   <X_NEMOService-Extension
xmlns="http://www.intertrust.com/discovery/upnp"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
   <nemoc:InspectionReference
xmlns:nemoc="http://nemo.intertrust.com/2004/core">
   <wsa:EndpointReference>
   <wsa:Address>
      http://www.intertrust.com/services/LicenseCreator.wsdl
   </wsa:Address>
   </wsa:EndpointReference>
   </nemoc:InspectionReference>
   </X_NEMOService-Extension>
</scpd>
```

6.5.9 Example SSDP Discovery Request Message

```
M-SEARCH * HTTP/1.1
S: uuid:ijklmnop-7dec-11d0-a765-00a0c91e6bf6
Host: 239.255.255.250:reservedSSDPport
Man: "ssdp:discover"
ST: http://nemo.intertrust.com/services#OctopusLicenseService
MX: 3
```

6.5.10 Example SSDP Discovery Response Message

```
HTTP/1.1 200 OK
S: uuid:ijklmnop-7dec-11d0-a765-00a0c91e6bf6
Ext:
Cache-Control: no-cache="Ext", max-age = 5000
ST: http://nemo.intertrust.com/services#OctopusLicenseService
USN: http://localhost:9082/MarlinSettop/services/OctopusLicense
Location: http://localhost:8080/service/inspection
Inspection-Policy-Id: urn:nemopolicyid:00001
```

7 References (Normative)

The following table lists references cited in this document.

[EXC-C14N]	*Exclusive XML Canonicalization*, W3C Recommendation, July 2002. http://www.w3.org/TR/xml-exc-c14n
[PKIX]	R. Housley *et al.*, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, RFC 3280, http://www.ietf.org/rfc/rfc3280.txt
[PKIXALGS]	W. Polk et al., *Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, RFC 3279, http://www.ietf.org/rfc/rfc3279.txt
[RFC1738]	Berners-Lee, T., Masinter, L., and M. McCahill, eds., *Uniform Resource Locators (URL)*, RFC 1738, December 1994, http://www.ietf.org/rfc/rfc1738.txt.
[RFC2141]	Moats, R., *URN Syntax*, RFC 2141, May 1997, http://www.ietf.org/rfc/rfc2141
[RFC2396]	T. Berners-Lee *et al.*, *Uniform Resource Identifiers (URI): Generic Syntax*, RFC 2396, August 1998, http://www.ietf.org/rfc/rfc2396.txt
[RFC2616]	R. Fielding *et al.*, *Hypertext Transfer Protocol – HTTP/1.1*, RFC 2616, June 1999, http://www.ietf.org/rfc/rfc2616.txt.
[RFC3280]	R. Housley *et al.*, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, RFC 3280, April 2002, http://www.ietf.org/rfc/rfc3280.txt
[SAML1.1]	Eve Maler, Prateek Mishra and Rob Philpott, eds., *Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V1.1*, http://www.oasis-open.org/committees/download.php/3406/oasis-%20sstc-saml-core-1.1.pdf
[SAML1.1Bind]	Eve Maler, Prateek Mishra and Rob Philpott, eds., *Bindings and Profiles for the OASIS Security Assertion Markup Language (SAML) V1.1*, http://www.oasis-open.org/committees/download.php/3405/oasis-%20sstc-saml-bindings-1.1.pdf.
[SAML2]	Scott Cantor, John Kemp and Eve Maler, eds., *Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0*, Working Draft 14, May 2004, http://www.oasis-open.org/committees/download.php/6998/sstc-saml-core-2.0-draft-14-diff.pdf

[SAML2Prof]	Frederick Hirsch, ed., *Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0*, OASIS Draft 6, July 2004, http://www.oasis-open.org/committees/download.php/7259/sstc-saml-profiles-2.0-draft-12-diff.pdf.
[SOAP 1.1]	Don Box et al. *Simple Object Access Protocol (SOAP) 1.1*, W3C Note, 8 May 2000. http://www.w3.org/TR/SOAP/
[SSDP]	*Simple Service Discovery Protocol - Version 1.0*. Internet Engineering Task Force, 1999-2000. http://www.upnp.org/download/draft_cai_ssdp_v1_03.txt
[TLS]	T. Dierks and C. Allen, *The TLS Protocol: Version 1.0*, IETF RFC 2246. http://www.ietf.org/rfc/rfc2246.txt
[UDDI]	*Universal Description, Discovery and Integration*. http://www.oasis-open.org/committees/uddi-spec/doc/contribs.htm#uddiv1
[UPnP]	*UPnP Device Architecture v1.0*. http://upnp.org/resources/documents/CleanUPnPDA101-20031202s.pdf
[WAPCertProf]	*WAP Certificate Profile Specification*, WAP Forum™, WAP-211-WAPCert-20010522-a http://www.openmobilealliance.org
[WS-ADDR]	M. Gudgin, M. Hadley, eds. *Web Services Addressing 1.0* - Core W3C Candidate Recommendation, 17 August 2005. http://www.w3.org/TR/2005/CR-ws-addr-core-20050817/
[WS-Discovery]	J. Beatty *et al*. *Web Services Dynamic Discovery* (WS-Discovery). October 2004. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/ws-discovery.asp
[WSDL 1.1]	E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. *Web Services Description Language (WSDL) 1.1*, W3C Note, March 2001. http://www.w3.org/TR/wsdl
[WSIBasicProfile11]	*Basic Profile Version 1.1*, August 24, 2004. http://www.ws-i.org/Profiles/BasicProfile-1.1.html
[WSIBasicSecurityProfile10]	*Basic Security Profile Version 1.0*, Working Group Draft, May 15, 2005. http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0-2005-05-15.html
[WSICCAM]	*Conformance Claim Attachment Mechanisms Version 1.0*, November 15, 2004. http://www.ws-i.org/Profiles/ConformanceClaims-1.0.html
[WSISOAPBinding]	*Simple SOAP Binding Profile Version 1.0*, August 24, 2004. http://www.ws-i.org/Profiles/SimpleSoapBindingProfile-1.0.html

[WS-MetadataExchange]	Keith Ballinger, et al. *Web Services Metadata Exchange* (WS-MetadataExchange), September 2004. http://msdn.microsoft.com/library/en-us/dnglobspec/html/ws-metadataexchange.pdf
[WS-POL]	*Web Services Policy Framework (WS-Policy)*, Version 1.1, 28 May 2003. http://www-106.ibm.com/developerworks/library/ws-polfram/
[WS-POL-ASSRT]	*Web Services Policy Assertions Language (WS-PolicyAssertions), Version 1.1*, 28 May 2003. http://www-106.ibm.com/developerworks/library/ws-polas/
[WS-POL-ATTCH]	*Web Services Policy Attachment (WS-PolicyAttachment), Version 1.1*, 28 May 2003. http://www-106.ibm.com/developerworks/library/ws-polatt/
[WS-SCON]	*Web Services Secure Conversation Language (WS-SecureConversation)*, Version 1.1, May 2004. http://www-106.ibm.com/developerworks/library/specification/ws-secon/
[WS-SEC]	*Web Services Security (WS-Security), Version 1.0*, OASIS, April 5, 2002. http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf
[WS-SEC-POL]	*Web Services Security Policy Language (WS-SecurityPolicy)*, Version 1.0, December 18, 2002. http://www-106.ibm.com/developerworks/library/ws-secpol/
[WS-SEC-SAML]	*Web Services Security: SAML Token Profile*, OASIS Standard, 1 December 2004. http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.0.pdf
[WS-SECX509]	Phillip Hallam-Baker *et al.*, eds. *Web Services Security X.509 Certificate Token Profile*, OASIS Standard 200401, March 2004, http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0.pdf
[WS-SECX509-ERR]	*Web Services Security: X.509 Token Profile 1.0 Errata 1.0*, OASIS Committee Draft 200512, December 2005. http://www.oasis-open.org/committees/download.php/16796/oasis-200512x509-token-profile-1.0-errata-005.pdf
[WS-TRUST]	*Web Services Trust Language (WS-Trust)*, May, 2004. ftp://www6.software.ibm.com/software/developer/library/ws-trust.pdf
[X.520]	*ITU-T Rec. X.520 (02/2001) Information technology – Open Systems Interconnection – The Directory: Selected attribute types*, http://www.itu.int/rec/recommendation.asp?type=items&lang=e&parent=T-REC-X.520-200102-I

[XML Names]	Tim Bray, Dave Hollander, and Andrew Layman (editors). *Namespaces in XML*. Textuality, Hewlett-Packard, and Microsoft. World Wide Web Consortium, 1999. http://www.w3.org/TR/REC-xml-names/
[XML Proto]	*XML Protocol*. http://www.w3.org/2000/xp/Group/
[XML-1.0]	Tim Bray, Jean Paoli, C.M. Sperberg-McQueen, Eve Maler, François Yergeau (editors). *Extensible Markup Language (XML) 1.0 (Third Edition)*. W3C Recommendation, February 2004. http://www.w3.org/TR/REC-xml/
[XMLBASE]	*XML Base*, W3C Recommendation, 27 June 2001. http://www.w3.org/TR/2001/REC-xmlbase-20010627/
[XMLDSIG]	Donald Eastlake, Joseph Reagle, and David Solo, eds. *XML-Signature Syntax and Processing*, W3C Recommendation, February 2002. http://www.w3.org/TR/xmldsig-core/
[XMLDSIG-MORE]	D. Eastlake 3rd, *Additional XML Security Uniform Resource Identifiers (URIs)*, RFC4051, April 2005, http://www.ietf.org/rfc/rfc4051.txt
[XMLENC]	D. Eastlake and J. Reagle, eds. *XML Encryption Syntax and Processing*, W3C Recommendation, December 2002. http://www.w3.org/TR/xmlenc-core/
[XPOINTER]	*XPointer Framework*, W3C Recommendation, 25 March 2003. http://www.w3.org/TR/2003/REC-xptr-framework-20030325/