

1  
2  
3  
4  
5  
6

# NEMO Technology Platform Specifications

7  
8 Version 1.1.1  
9 Final

10  
11  
12  
13  
14  
15  
16

Source  
Date

Marlin Developer Community  
February 5, 2010

17

18 **Notice**

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

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

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

40 Copyright © 2003 - 2010 by MDC, 415-112 North Mary Avenue #383 Sunnyvale,  
41 CA 94085, USA. All rights reserved. Third-party brands and names are the  
42 property of their respective owners.

43 **Intellectual Property**

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

46 **Contact Information**

47 Feedback on this specification should be addressed to:  
48 [editor@marlin-community.com](mailto:editor@marlin-community.com)

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

51

# Table of Contents

53	<b>1 Introduction .....</b>	<b>6</b>
54	<b>1.1 Namespaces .....</b>	<b>7</b>
55	<b>1.2 Notation .....</b>	<b>7</b>
56	<b>2 NEMO Message Bindings .....</b>	<b>9</b>
57	<b>2.1 Overview .....</b>	<b>9</b>
58	<b>2.2 SOAP Binding .....</b>	<b>9</b>
59	2.2.1 FaultDetails Header Element .....	9
60	2.2.2 wsa:Action and soapAction .....	9
61	<b>2.3 Message Correlation .....</b>	<b>9</b>
62	2.3.1 Example .....	10
63	<b>2.4 Service Description Bindings .....</b>	<b>11</b>
64	2.4.1 SOAP Header Faults .....	12
65	<b>3 NEMO Security Bindings .....</b>	<b>13</b>
66	<b>3.1 WS-Security Binding .....</b>	<b>13</b>
67	3.1.1 Overview .....	13
68	3.1.2 NEMO Secure Messaging Protocol Elements .....	13
69	3.1.3 Message Security Elements .....	13
70	3.1.3.1 Timestamp .....	13
71	3.1.3.2 Nonce .....	13
72	3.1.3.3 Digital Signature .....	13
73	3.1.3.4 Encryption .....	14
74	3.1.3.5 NEMO-Specific Elements .....	15
75	3.1.4 Basic Secure Messaging Protocol .....	19
76	3.1.4.1 Full Security .....	19
77	3.1.4.2 Other Protection Levels .....	29
78	3.1.5 Recommended Fault Codes .....	30
79	3.1.6 Examples .....	31
80	3.1.6.1 Basic Secure Messaging Protocol .....	31
81	<b>3.2 Security Considerations .....</b>	<b>47</b>
82	<b>4 NEMO Trust Management Bindings .....</b>	<b>48</b>
83	<b>4.1 Introduction (Informative) .....</b>	<b>48</b>
84	<b>4.2 URI NEMO Identifier Binding .....</b>	<b>48</b>
85	4.2.1 Canonical Representation of HTTP URLs .....	48
86	4.2.2 Canonical Representation of URNs .....	49
87	<b>4.3 X.509 Authentication Binding .....</b>	<b>49</b>
88	4.3.1 X.509 Object Identifiers .....	49
89	4.3.1.1 NEMO .....	49
90	4.3.1.2 Name Attributes .....	49
91	4.3.1.3 Extended Key Usages .....	49
92	4.3.2 X.509 Certificate Use .....	50
93	4.3.3 Signature Algorithms .....	50

94	4.3.4 Distinguished Names.....	50
95	4.3.5 Certificate Revocation.....	50
96	4.3.6 Certificate and Key Renewal.....	50
97	<b>4.4 SAML NEMO Node Attribute Assertions .....</b>	<b>51</b>
98	4.4.1 Processing Rules for SAML Attributes.....	51
99	4.4.1.1 Attribute Association.....	51
100	4.4.1.2 SAML Conditions .....	52
101	4.4.1.3 Validity Caching.....	52
102	4.4.2 NEMO Role Attributes .....	52
103	4.4.3 NEMO Attribute Reliance Policy (Informative) .....	52
104	<b>4.5 SSL/TLS Service Authorization.....</b>	<b>52</b>
105	4.5.1 Certificate Revocation.....	53
106	4.5.2 Certificate and Key Renewal.....	53
107	<b>4.6 Annex.....</b>	<b>54</b>
108	<b>5 NEMO Policy Bindings.....</b>	<b>55</b>
109	<b>5.1 Overview .....</b>	<b>55</b>
110	<b>5.2 NEMO Web Service Policy Binding.....</b>	<b>55</b>
111	5.2.1 Protocol Policy.....	56
112	5.2.1.1 Security Tokens.....	56
113	5.2.1.2 nemop:Usage Attribute.....	56
114	5.2.1.3 nemosec:Usage Attribute.....	56
115	5.2.1.4 MessageAge .....	56
116	5.2.1.5 Nonce .....	56
117	5.2.1.6 Signed Message Key .....	57
118	5.2.1.7 Signed Challenge / Signed Challenge Response .....	57
119	5.2.1.8 Message Integrity .....	58
120	5.2.1.9 Message Confidentiality .....	58
121	5.2.1.10Message Parts.....	59
122	5.2.1.11XML Element Encryption Policy .....	59
123	5.2.2 Message Security Policy .....	60
124	5.2.2.1 Protocol Assertion.....	60
125	5.2.2.2 Profile Assertion.....	61
126	5.2.3 Application Security Policy.....	61
127	5.2.3.1 SAML Attribute Assertion Token .....	61
128	5.2.4 Session Policy .....	63
129	5.2.4.1 Session Duration .....	63
130	5.2.5 Caching Policy .....	64
131	5.2.5.1 Signaling Credential Caching.....	64
132	5.2.6 Examples.....	64
133	5.2.6.1 Basic Protocol – No Sessions.....	64
134	5.2.6.2 Secure Conversation Protocol .....	70
135	<b>6 NEMO Discovery/Inspection Bindings .....</b>	<b>79</b>
136	<b>6.1 Overview .....</b>	<b>79</b>
137	<b>6.2 Service Discovery Binding.....</b>	<b>79</b>
138	6.2.1 Overview.....	79
139	6.2.2 WS-Discovery.....	79
140	6.2.2.1 WS-Discovery Announcement .....	79
141	6.2.2.2 WS-Discovery Discovery Proxy .....	80
142	6.2.2.3 Matching Criteria .....	81

143	6.2.2.4 Service References, Properties and Parameters.....	82
144	6.2.3 SSDP-based Discovery .....	83
145	6.2.3.1 Goals .....	83
146	6.2.3.2 Discovery Protocol Support .....	83
147	6.2.3.3 Service Identifier .....	84
148	6.2.3.4 Service Inspection .....	84
149	6.2.3.5 Inspection Policy Identifier .....	84
150	6.2.4 Presence Announcements.....	85
151	6.2.5 Additional UPnP-Related Extensions.....	85
152	6.2.5.1 Extensions to Device Description .....	86
153	6.2.5.2 Extensions to Service Description.....	86
154	6.2.5.3 UPnP Service Security .....	87
155	<b>6.3 Inspection Binding .....</b>	<b>87</b>
156	6.3.1 Description.....	87
157	6.3.2 Requirements (Normative).....	88
158	<b>6.4 Schema and Abstract Web Service Definitions .....</b>	<b>88</b>
159	6.4.1 NEMO Node Description.....	88
160	6.4.2 Discovery .....	89
161	6.4.2.1 WS-Discovery Scope-Related Definitions .....	89
162	6.4.2.2 WS-Discovery Discovery Proxy Schema Definitions and WSDL .....	90
163	6.4.3 Inspection.....	91
164	6.4.3.1 NEMO UPnP Device Description Extension .....	91
165	6.4.3.2 NEMO UPnP Service Description Extension .....	92
166	<b>6.5 Sample Messages .....</b>	<b>92</b>
167	6.5.1 WS-Discovery Probe for a Service that Supports DiscoveryProxy.....	92
168	6.5.2 WS-Discovery ProbeMatch for a Service that Supports DiscoveryProxy.....	93
169	6.5.3 WS-Discovery Hello Announcement to Announce Availability of Service .....	95
170	6.5.4 WS-Discovery Bye to Announce Service Is No Longer Available .....	96
171	6.5.5 WS-MetadataExchange Request to Obtain a Service WSDL .....	97
172	6.5.6 WS-MetadataExchange Response for Obtained Service WSDL.....	98
173	6.5.7 Example UPnP Device Description with NEMO Extension .....	99
174	6.5.8 Example UPnP Service Description with NEMO Extension .....	101
175	6.5.9 Example SSDP Discovery Request Message .....	102
176	6.5.10 Example SSDP Discovery Response Message.....	102
177	<b>7 References (Normative) .....</b>	<b>103</b>
178		

# 179 1 Introduction

180 This document contains the NEMO technology platform specifications.

181 Please note: This document consolidates material that was previously in separate documents. The  
182 highest version number among the source documents consolidated into this document was 1.1.  
183 The version number for this document is initially one greater than that in the third digit, that is,  
184 1.1.1.

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

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

193 1. NEMO Message Bindings (§2)

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

196 2. NEMO Security Bindings (§3)

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

199 3. NEMO Trust Management Bindings (§4)

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

204 4. NEMO Policy Bindings (§5)

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

207 5. NEMO Discovery/Inspection Bindings (§6)

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

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

## 216 1.1 Namespaces

217 The following namespaces are used in this document:

218

219

220

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

## 221 1.2 Notation

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

224 `&nemo;/Element`

225 corresponds to

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

227 and

228 `&nemop;/Element`

229 corresponds to

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

231 These entities are used as shorthand notation throughout this document.

232

## 233 2 NEMO Message Bindings

### 234 2.1 Overview

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

### 237 2.2 SOAP Binding

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

243 NEMO messages SHALL comply with [\[SOAP 1.1\]](#) and with the Web Services-Interoperability  
244 Organization Basic Profile 1.1 [\[WSIBasicProfile11\]](#) and Simple SOAP Binding Profile  
245 [\[WSISOAPBinding\]](#).

#### 246 2.2.1 FaultDetails Header Element

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

#### 251 2.2.2 wsa:Action and soapAction

252 Every NEMO SOAP message document MUST contain a `<wsa:Action>` element compliant  
253 with [\[WS-ADDR\]](#).

### 254 2.3 Message Correlation

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

260 This specification defines a WS-Addressing relationship type:

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

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

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

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

268 Three OPTIONAL correlation usage patterns are described below.

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

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

## 284 2.3.1 Example

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

```
288  
289 <!-- Request message -->  
290 <soap:Envelope  
291   xmlns:soap="http://www.w3.org/2003/05/soap-envelope"  
292   xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"  
293   xmlns:nemoc="http://nemo.intertrust.com/2004/core"  
294   xmlns:tns="http://example.com/myNamespace">  
295   <soap:Header>  
296     <wsa:To>http://example.com/myService</wsa:To>  
297     <wsa:Action>http://example.com/myService/myOperation</wsa:Action>  
298     <wsa:ReplyTo>  
299       <wsa:Address>http://fabrikam.com/client</wsa:Address>  
300     </wsa:ReplyTo>  
301     <wsa:MessageID>uuid:aaaabbbb-cccc-0001</wsa:MessageID>  
302     <wsa:RelatesTo  
303 RelationshipType="http://nemo.intertrust.com/2004/addressing/originatesF  
304 rom"> uuid:aaaabbbb-cccc-0001</wsa:RelatesTo>  
305   </soap:Header>  
306   <soap:Body/>  
307 </soap:Envelope>
```

308

```

309 <!-- Response message -->
310 <soap:Envelope
311   xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
312   xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
313   xmlns:nemoc="http://nemo.intertrust.com/2004/core"
314   xmlns:tns="http://example.com/myNamespace">
315   <soap:Header>
316     <wsa:To>http://fabrikam.com/client</wsa:To>
317     <wsa:Action>http://example.com/myNamespace/myPortType/
318 myOperationResponse</wsa:Action>
319     <wsa:ReplyTo>
320       <wsa:Address>http://example.com/myService</wsa:Address>
321     </wsa:ReplyTo>
322     <wsa:MessageID>uuid:aaaabbbb-cccc-0002</wsa:MessageID>
323     <wsa:RelatesTo
324 RelationshipType="http://nemo.intertrust.com/2004/addressing/originatesF
325 rom"> uuid:aaaabbbb-cccc-0001</wsa:RelatesTo>
326     <wsa:RelatesTo
327 RelationshipType="http://www.w3.org/2005/08/addressing/reply">uuid:aaaab
328 bbb-cccc-0001</wsa:RelatesTo>
329   </soap:Header>
330   <soap:Body/>
331 </soap:Envelope>

```

```

332 <!-- Confirmation message -->
333 <soap:Envelope
334   xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
335   xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
336   xmlns:nemoc="http://nemo.intertrust.com/2004/core"
337   xmlns:tns="http://example.com/myNamespace">
338   <soap:Header>
339     <wsa:To>http://example.com/myService</wsa:To>
340     <wsa:Action>http://example.com/myNamespace/myPortType/
341 myOperationConfirmation</wsa:Action>
342     <wsa:MessageID>uuid:aaaabbbb-cccc-0003</wsa:MessageID>
343     <wsa:RelatesTo
344 RelationshipType="http://nemo.intertrust.com/2004/addressing/originatesF
345 rom"> uuid:aaaabbbb-cccc-0001</wsa:RelatesTo>
346     <wsa:RelatesTo
347 RelationshipType="http://www.w3.org/2005/08/addressing/reply">uuid:aaaab
348 bbb-cccc-0002</wsa:RelatesTo>
349   </soap:Header>
350   <soap:Body/>
351 </soap:Envelope>

```

## 353 **2.4 Service Description Bindings**

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

358 NEMO services SHALL be described using service descriptions as specified in [\[WSDL 1.1\]](#).

### 359 **2.4.1 SOAP Header Faults**

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

## 362 **3 NEMO Security Bindings**

### 363 **3.1 WS-Security Binding**

#### 364 **3.1.1 Overview**

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

#### 369 **3.1.2 NEMO Secure Messaging Protocol Elements**

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

- 373 • Timestamps
- 374 • Nonces
- 375 • Symmetric keys
- 376 • Asymmetric keys
- 377 • Encrypted data
- 378 • Digital signatures

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

#### 381 **3.1.3 Message Security Elements**

##### 382 **3.1.3.1 Timestamp**

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

##### 386 **3.1.3.2 Nonce**

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

##### 388 **3.1.3.3 Digital Signature**

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

### 393 3.1.3.3.1 Canonicalization Algorithm

394 The signatures SHALL use exclusive canonicalization, as defined by [\[EXC-C14N\]](#) and  
395 recommended by [\[WS-SEC\]](#) and [\[WSIBasicSecurityProfile10\]](#).

### 396 3.1.3.3.2 Signing Algorithm

397 Message integrity for the SOAP Message Binding SHALL be provided with XML Signature  
398 [\[XMLDSIG\]](#). One of the following algorithms SHALL be used [\[XMLDSIG\]](#), [\[XMLDSIG-](#)  
399 [MORE\]](#).

400

Purpose	Name	URI
Asymmetric Signature	RSA-SHA256	<a href="http://www.w3.org/2001/04/xmlsig-more#rsa-sha256">http://www.w3.org/2001/04/xmlsig-more#rsa-sha256</a>

### 401 3.1.3.3.3 Message Digest Algorithm

402 XML Signatures SHALL digest referenced elements using one of the following algorithms. See  
403 [\[XMLDSIG\]](#), [\[XMLDSIG-MORE\]](#), [\[XML-ENC\]](#).

404

Purpose	Name	URI
Digest within Signature	SHA256	<a href="http://www.w3.org/2001/04/xmlenc#sha256">http://www.w3.org/2001/04/xmlenc#sha256</a>

### 405 3.1.3.4 Encryption

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

### 413 3.1.3.4.1 Encryption Algorithms

414 Message confidentiality for the SOAP Message Binding SHALL be provided with XML  
415 Encryption [\[XMLENC\]](#). The following algorithms SHALL be used:

416

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

### 417 **3.1.3.5 NEMO-Specific Elements**

#### 418 **3.1.3.5.1 Usage Attribute**

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

#### 427 **3.1.3.5.2 Receiver NEMO Node**

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

##### 431 **3.1.3.5.2.1 Syntax**

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

433  
434  
435  
436

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

437 `.../ToNode`

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

441 `.../ToNode/@{any}`

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

444 `.../ToNode/{any}`

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

#### 447 **3.1.3.5.3 Sender NEMO Node**

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

##### 452 **3.1.3.5.3.1 Syntax**

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

454

```
455 <FromNode wsu:Id="...">
456     urn:nemo:node:...
457 </FromNode>
```

458 .../FromNode

459 The string content of this element is the canonicalized URI identifier of the original sending  
460 NEMO node, in UTF-8 encoding, as specified by NEMO Trust Management Bindings in  
461 §4.2.

462 .../FromNode/@{any}

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

465 .../FromNode/{any}

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

### 468 3.1.3.5.4 Protocol Declaration

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

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

#### 482 3.1.3.5.4.1 Syntax

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

```
484
485 <ProtocolDeclaration wsu:Id="..." URI="..." nemosec:Usage="...">
486     <Step index="..." Type="..." />
487     <Reference nemosec:TargetUsage="..." URI="..." />
488 </ProtocolDeclaration>
```

489 .../ProtocolDeclaration

490 An element identifying a messaging protocol.

491 .../ProtocolDeclaration/@URI

492 The URI identifier of the protocol.

493 .../ProtocolDeclaration/Step

494 An element identifying a step in a messaging protocol.

495 `.../ProtocolDeclaration/Step/@Index`

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

498 `.../ProtocolDeclaration/Step/@Type`

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

505 `fault`

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

512 `.../ProtocolDeclaration/Step/@{any}`

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

515 `.../ProtocolDeclaration/Reference`

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

518 `.../ProtocolDeclaration/Reference/@nemosec:TargetUsage`

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

520 `.../ProtocolDeclaration/Reference/@URI`

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

523 `.../ProtocolDeclaration/@{any}`

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

526 `.../ProtocolDeclaration/{any}`

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

530 **3.1.3.5.5 Correspondence of WSDL 1.1 Messages and Basic Secure**  
531 **Messaging Protocol Step Types**

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

538 **3.1.3.5.6 Profile**

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

544 `&nemosec;/profile`

545 **3.1.3.5.6.1 Syntax**

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

547 `<Profile wsu:Id="..." URI="..." />`

548 `.../Profile`

549 An element defining a NEMO profile.

550 `.../Profile/@URI`

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

552 `.../Profile/@{any}`

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

555 `.../Profile/{any}`

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

558 **3.1.3.5.7 BinarySecurityToken ValueTypes**

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

561

Symmetric Key	<code>&amp;nemosec;/BST/SymmetricKey</code>
---------------	---

562 **3.1.3.5.8 Role Assertions**

563 The NEMO Trust Management Binding section (§4) defines the use of SAML 1.1 attribute  
564 assertions to assert NEMO node roles. According to [\[WS-SEC-SAML\]](#), SAML assertions may

565 be placed within a `<wsse:Security>` element, and may be referenced from a  
566 `<wsse:SecurityTokenReference>` element.

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

569 `&nemo;/attribute/role`

### 570 **3.1.3.5.9 Fault Codes**

571 Messages containing faults MAY use any fault codes defined in [\[SOAP 1.1\]](#) or [\[WS-SEC\]](#). The  
572 following SOAP fault codes MAY also be returned in a `<faultcode>` element within an  
573 `<S11:Fault>` element, as specified in [\[SOAP 1.1\]](#). The fault codes are defined in the  
574 `nemosec: namespace`.  
575

Name	Meaning
UnsupportedSecureProtocol	A message contained a <code>&lt;nemosec:ProtocolDeclaration&gt;</code> that specifies an unsupported secure protocol.

## 576 **3.1.4 Basic Secure Messaging Protocol**

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

### 581 **3.1.4.1 Full Security**

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

#### 589 **3.1.4.1.1 Request Message**

##### 590 **3.1.4.1.1.1 Security Header**

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

598 **3.1.4.1.1.1 Protocol Identifier**

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

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

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

604 `request`

605 **3.1.4.1.1.2 Requestor's Timestamp**

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

611 `#request-timestamp`

612 **3.1.4.1.1.3 Requestor's Nonce**

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

618 `#request-nonce`

619 **3.1.4.1.1.4 Requestor's Identifier**

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

625 `#request-fromNode`

626 **3.1.4.1.1.5 Responder's Identifier**

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

632 `#request-toNode`

633 **3.1.4.1.1.6 Self-encrypted Message Key**

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

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

641 #request-messageKey

#### 642 **3.1.4.1.1.7 Requestor's Public Signing Key Certificates**

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

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

647 or a ValueType attribute of

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

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

657 #request-signingKey

#### 658 **3.1.4.1.1.8 Requestor's Public Encryption Key Certificates**

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

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

663 or a ValueType attribute of

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

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

673 #response-encryptionKey

674 **3.1.4.1.1.9 Requestor's Encrypted Message Encryption Key**

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

681 #request-encryptedMessageKey

682 **3.1.4.1.1.10 Signature**

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

698 #request-signature

699 **3.1.4.1.1.2 Processing Rules**

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

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

715 The responder SHALL verify the signature in the request message, the responder SHALL  
716 validate the binding between the requestor's node ID and the signature key, and the responder  
717 SHALL verify that the signature signs at least the parts specified in §3.1.4.1.1.10. If the

718 security header contains a `<nemosec:FromNode>` element, the responder SHALL verify that  
719 the element contains the requestor's node ID.

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

## 722 **3.1.4.1.2 Response Message**

### 723 **3.1.4.1.2.1 Security Header**

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

#### 730 **3.1.4.1.2.1.1 Protocol Identifier**

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

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

736 `response`

#### 737 **3.1.4.1.2.1.2 Responder's Timestamp**

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

743 `#response-timestamp`

#### 744 **3.1.4.1.2.1.3 Responder's Identifier**

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

750 `#response-fromNode`

#### 751 **3.1.4.1.2.1.4 Requestor's Identifier**

752 The `<wsse:Security>` element SHALL contain a `<nemosec:ToNode>` element  
753 containing the NEMO node ID of the requestor. The `<nemosec:ToNode>` element MAY  
754 contain a `nemosec:Usage` attribute with a value containing the value of the URI attribute in

755 the <nemosec:ProtocolDeclaration> with the following fragment concatenated at the  
756 end:

757 #response-toNode

#### 758 **3.1.4.1.2.1.5 Self-encrypted Message Key**

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

766 #response-messageKey

#### 767 **3.1.4.1.2.1.6 Requestor's Nonce**

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

774 #response-returnedNonce

#### 775 **3.1.4.1.2.1.7 Responder's Nonce**

776 If a later confirmation message is expected from the requestor, the <wsse:Security>  
777 element SHALL contain a <wsse:Nonce> with a ValueType attribute as specified in  
778 §3.1.3.2 containing a base64-encoded octet string of no more than 128 octets generated by the  
779 responder. The <wsse:Nonce> element MAY contain a nemosec:Usage attribute with a  
780 value containing the value of the URI attribute in the <nemosec:ProtocolDeclaration>  
781 with the following fragment concatenated at the end:

782 #response-nonce

#### 783 **3.1.4.1.2.1.8 Responder's Public Signing Key Certificates**

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

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

788 or a ValueType attribute of

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

791 [\[WS-SECX509\]](#) containing the responder's long-term public signing key and associated  
792 certificates. At least one included certificate SHALL have a subject name containing the NEMO  
793 node ID of the requestor and the transmitted subject public key. For transmission of certificate  
794 chains using PKIPath, the last certificate in the chain SHALL NOT be signed by the certificate's

795 subject public key pair. The <wsse:BinarySecurityToken> element MAY contain a  
796 nemosec:Usage attribute with a value containing the value of the URI attribute in the  
797 <nemosec:ProtocolDeclaration> with the following fragment concatenated at the end:

798 #response-signingKey

#### 799 **3.1.4.1.2.1.9 Responder's Encrypted Message Encryption Key**

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

806 #response-encryptedMessageKey

#### 807 **3.1.4.1.2.1.10 Signature**

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

822 #response-signature

#### 823 **3.1.4.1.2.2 Processing Rules**

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

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

838 check that the unencrypted responder's message encryption key that appeared in the message  
839 signature is the same as is used to encrypt the message.

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

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

### 847 **3.1.4.1.3 Confirmation Message**

#### 848 **3.1.4.1.3.1 Security Header**

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

##### 856 **3.1.4.1.3.1.1 Protocol Identifier**

857 The <wsse:Security> element SHALL include a  
858 <nemosec:ProtocolDeclaration> element with a URI having the value specified in  
859 §3.1.4.1.1.1.1.

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

862 `confirmation`

##### 863 **3.1.4.1.3.1.2 Requestor's Timestamp**

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

869 `#confirmation-timestamp`

##### 870 **3.1.4.1.3.1.3 Responder's Nonce**

871 The <wsse:Security> element SHALL contain a <wsse:Nonce>, as specified in  
872 §3.1.3.2, containing the responder's nonce passed in the corresponding response message. The  
873 <wsse:Nonce> element MAY contain a nemosec:Usage attribute with a value containing  
874 the value of the URI attribute in the <nemosec:ProtocolDeclaration> with the  
875 following fragment concatenated at the end:

876 `#confirmation-returnedNonce`

877 **3.1.4.1.3.1.4 Requestor's Identifier**

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

883 #confirmation-fromNode

884 **3.1.4.1.3.1.5 Responder's Identifier**

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

890 #confirmation-toNode

891 **3.1.4.1.3.1.6 Self-encrypted Message Key**

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

899 #confirmation-messageKey

900 **3.1.4.1.3.1.7 Requestor's Encrypted Message Encryption Key**

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

906 #confirmation-encryptedMessageKey

907 **3.1.4.1.3.1.8 Requestor's Public Signing Key Certificates**

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

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

912 or a ValueType attribute of

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

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

922 #confirmation-signingKey

#### 923 **3.1.4.1.3.1.9 Signature**

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

937 #confirmation-signature

#### 938 **3.1.4.1.3.2 Processing Rules**

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

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

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

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

#### 961 **3.1.4.1.4 Fault Response Message**

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

967 `request-fault`

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

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

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

987 `#fault-toNode`

#### 988 **3.1.4.1.5 Responder’s Public Encryption Key**

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

993 `#request-encryptionKey`

#### 994 **3.1.4.2 Other Protection Levels**

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

997 **3.1.4.2.1 Freshness Protections**

998 If timestamp and nonce elements are not to be used within a message, they SHOULD NOT  
999 appear in the message. The processing rules SHALL remain the same as for the full security  
1000 case, except for the checking of nonces and/or timestamps.

1001 Note: If nonces are used without timestamps, this specification does not specify how long  
1002 receiver nodes should remember the nonces that have been used to date.

1003 **3.1.4.2.2 Integrity Protection**

1004 If integrity protection is not to be used within a message, the signature element and the sender's  
1005 certificates SHOULD NOT appear in the <wsse:Security> element. The processing rules  
1006 SHALL remain the same as for the full security case, except for the validating of the signature  
1007 and the binding of the signing key to the sender's node ID. If integrity protection is not used, a  
1008 message SHOULD contain the sender's NEMO node identifier in a <nemosec:FromNode>  
1009 element.

1010 **3.1.4.2.3 Confidentiality Protection**

1011 If confidentiality protection is not to be used within a message, the message payload SHALL  
1012 NOT be encrypted. The processing rules SHALL remain the same as for the full security case,  
1013 except for the decrypting of the received message payload, and validating the binding between  
1014 the receiver's node ID and public encryption key.

1015 **3.1.5 Recommended Fault Codes**

1016 The following fault codes are RECOMMENDED for use in the described situations.  
1017

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

1018 **3.1.6 Examples**

1019 **3.1.6.1 Basic Secure Messaging Protocol**

1020 **3.1.6.1.1 Request Message**

1021

1022

1023

1024

1025

1026

1027

1028

1029

1030

1031

1032

1033

1034

1035

1036

1037

1038

1039

1040

1041

1042

1043

1044

1045

1046

1047

1048

1049

1050

1051

1052

```
<?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
```

```

1053
1054     nemosec:TargetUsage="http://nemo.intertrust.com/2005/10/security/secu
1055 re-protocol/basic/1.0#request-signature"
1056         URI="#NEMO_ID3"/>
1057     </nemosec:ProtocolDeclaration>
1058
1059     <!-- Timestamp -->
1060     <wsu:Timestamp
1061         wsu:Id="NEMO_ID6"
1062     >
1063         nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
1064 protocol/basic/1.0#request-timestamp">
1065             <wsu:Created>2005-06-15T02:08:21.372</wsu:Created>
1066             <wsu:Expires>2005-06-15T03:08:21.372</wsu:Expires>
1067         </wsu:Timestamp>
1068
1069     <!-- Nonce -->
1070     <wsse:Nonce
1071         wsu:Id="NEMO_ID7"
1072     >
1073         nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
1074 protocol/basic/1.0#request-nonce"
1075         >UU4oOCriFXA=</wsse:Nonce>
1076
1077     <!-- ToNode -->
1078     <nemosec:ToNode
1079         wsu:Id="NEMO_ID8"
1080     >
1081         nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
1082 protocol/basic/1.0#request-toNode"
1083         >urn:nemo:node:ObjectProvider</nemosec:ToNode>
1084
1085     <!-- FromNode -->
1086     <nemosec:FromNode
1087         wsu:Id="NEMO_ID9"
1088     >
1089         nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
1090 protocol/basic/1.0#request-fromNode"
1091
1092         >1.3.6.1.4.1.7584.1.1.1=urn:nemo:node:Device</nemosec:FromNode>
1093
1094     <!-- Encrypted message key -->
1095     <enc:EncryptedKey
1096         Id="NEMO_ID1">
1097         <enc:EncryptionMethod
1098             Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-
1099 mgf1p"/>
1100         <enc:CipherData>
1101             <enc:CipherValue>wboB...Ii4=</enc:CipherValue>
1102         </enc:CipherData>
1103         <enc:ReferenceList>
1104             <!-- Encrypted signature element -->

```

```

1105         <enc:DataReference URI="#NEMO_ID15"/>
1106         <!-- Encrypted body contents -->
1107         <enc:DataReference URI="#NEMO_ID16"/>
1108     </enc:ReferenceList>
1109 </enc:EncryptedKey>
1110
1111     <!-- Self-encrypted message key -->
1112     <wsse:BinarySecurityToken
1113
1114         ValueType="http://nemo.intertrust.com/2005/10/security/BST/SymmetricK
1115 ey"
1116         EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
1117 wss-soap-message-security-1.0#Base64Binary"
1118         wsu:Id="NEMO_ID10"
1119
1120         nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
1121 protocol/basic/1.0#request-messageKey"
1122         >/1KLw02aJ73ByZ1ya1ZFvw==</wsse:BinarySecurityToken>
1123
1124     <!-- Message signing certificate chain -->
1125     <wsse:BinarySecurityToken
1126         ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-
1127 200401-wss-x509-token-profile-1.0#X509PKIPathv1"
1128         EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
1129 wss-soap-message-security-1.0#Base64Binary"
1130         wsu:Id="NEMO_ID2"
1131
1132         nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
1133 protocol/basic/1.0#request-signingKey"
1134         >MIIC...xUM=</wsse:BinarySecurityToken>
1135
1136     <!-- Signature -->
1137     <ds:Signature
1138         Id="NEMO_ID3">
1139         <ds:SignedInfo>
1140             <ds:CanonicalizationMethod
1141 Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
1142                 <ec:InclusiveNamespaces/>
1143             </ds:CanonicalizationMethod>
1144             <ds:SignatureMethod
1145 Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
1146
1147             <!-- Protocol declaration -->
1148             <ds:Reference URI="#NEMO_ID4">
1149                 <ds:Transforms>
1150                     <ds:Transform
1151 Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
1152                         <ec:InclusiveNamespaces/>
1153                     </ds:Transform>
1154                 </ds:Transforms>
1155                 <ds:DigestMethod
1156 Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

```

```

1157
1158 <ds:DigestValue>nI8PbZSH68Hs85fnEhFu4tmbutQ=</ds:DigestValue>
1159 </ds:Reference>
1160
1161 <ds:DigestValue>c84XiUsfpEk9YUHeMFtyd71/wsw=</ds:DigestValue>
1162 </ds:Reference>
1163
1164 <!-- Timestamp -->
1165 <ds:Reference URI="#NEMO_ID6">
1166 <ds:Transforms>
1167 <ds:Transform
1168 Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
1169 <ec:InclusiveNamespaces/>
1170 </ds:Transform>
1171 </ds:Transforms>
1172 <ds:DigestMethod
1173 Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1174
1175 <ds:DigestValue>X/b8LS+q9YaLIiJbidJLLyzNsR4=</ds:DigestValue>
1176 </ds:Reference>
1177
1178 <!-- Nonce -->
1179 <ds:Reference URI="#NEMO_ID7">
1180 <ds:Transforms>
1181 <ds:Transform
1182 Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
1183 <ec:InclusiveNamespaces/>
1184 </ds:Transform>
1185 </ds:Transforms>
1186 <ds:DigestMethod
1187 Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1188
1189 <ds:DigestValue>96l29nkFs+ZYc8JHESTZRDXLlwo=</ds:DigestValue>
1190 </ds:Reference>
1191
1192 <!-- ToNode -->
1193 <ds:Reference URI="#NEMO_ID8">
1194 <ds:Transforms>
1195 <ds:Transform
1196 Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
1197 <ec:InclusiveNamespaces/>
1198 </ds:Transform>
1199 </ds:Transforms>
1200 <ds:DigestMethod
1201 Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1202
1203 <ds:DigestValue>FtoLVhdlmc3vY/v9DOEnj9408/I=</ds:DigestValue>
1204 </ds:Reference>
1205
1206 <!-- FromNode -->
1207 <ds:Reference URI="#NEMO_ID9">
1208 <ds:Transforms>

```

```

1209         <ds:Transform
1210 Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
1211         <ec:InclusiveNamespaces/>
1212         </ds:Transform>
1213     </ds:Transforms>
1214     <ds:DigestMethod
1215 Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1216
1217     <ds:DigestValue>Xh5nY1FqNoGA9RqJuBJN19Lg6G8=</ds:DigestValue>
1218     </ds:Reference>
1219
1220     <!-- Self-encrypted message key -->
1221     <ds:Reference URI="#NEMO_ID10">
1222         <ds:Transforms>
1223             <ds:Transform
1224 Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
1225             <ec:InclusiveNamespaces/>
1226             </ds:Transform>
1227         </ds:Transforms>
1228         <ds:DigestMethod
1229 Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1230
1231         <ds:DigestValue>LvALWu5ULqUaHRefe0NWQTNShBM=</ds:DigestValue>
1232         </ds:Reference>
1233
1234         <!-- Response encryption key -->
1235         <ds:Reference URI="#NEMO_ID13">
1236             <ds:Transforms>
1237                 <ds:Transform
1238 Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
1239                 <ec:InclusiveNamespaces/>
1240                 </ds:Transform>
1241             </ds:Transforms>
1242             <ds:DigestMethod
1243 Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1244
1245             <ds:DigestValue>4jkyhQYEulgsrocogUCHPsl1NwE=</ds:DigestValue>
1246             </ds:Reference>
1247
1248             <!-- SOAP Body -->
1249             <ds:Reference URI="#NEMO_ID14">
1250                 <ds:Transforms>
1251                     <ds:Transform
1252 Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
1253                     <ec:InclusiveNamespaces/>
1254                     </ds:Transform>
1255                 </ds:Transforms>
1256                 <ds:DigestMethod
1257 Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1258
1259                 <ds:DigestValue>oQM3cTcTn2WYP0F6ac6+dwWJnfA=</ds:DigestValue>
1260                 </ds:Reference>

```

```

1261     </ds:SignedInfo>
1262     <ds:SignatureValue>wBbi...jvA=</ds:SignatureValue>
1263     <ds:KeyInfo>
1264         <wsse:SecurityTokenReference>
1265             <!-- Message signing certificate chain -->
1266             <wsse:Reference URI="#NEMO_ID2"/>
1267         </wsse:SecurityTokenReference>
1268     </ds:KeyInfo>
1269 </ds:Signature>
1270
1271 <!-- Response encryption key certificate chain -->
1272 <wsse:BinarySecurityToken
1273     ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-
1274 200401-wss-x509-token-profile-1.0#X509PKIPathv1"
1275     EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
1276 wss-soap-message-security-1.0#Base64Binary"
1277     wsu:Id="NEMO_ID13"
1278
1279     nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
1280 protocol/basic/1.0#response-encryptionKey"
1281 >MIIC...Wgc=</wsse:BinarySecurityToken>
1282
1283 <!-- Role assertion -->
1284 <saml:Assertion
1285 xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
1286 AssertionID="ID1117583305016" IssueInstant="2005-05-31T23:48:24Z"
1287 Issuer="1.3.6.1.4.1.7584.1.1.1=urn:nemo:node:ObjectAuthority"
1288 MajorVersion="1" MinorVersion="1">
1289     <saml:Conditions NotBefore="2005-05-31T23:48:24Z"
1290 NotOnOrAfter="2006-05-31T23:48:24Z"/>
1291     <saml:AttributeStatement>
1292         <saml:Subject>
1293             <saml:NameIdentifier
1294 Format="urn:oasis:names:tc:SAML:1.1:nameid-
1295 format:X509SubjectName">1.3.6.1.4.1.7584.1.1.1=urn:nemo:node:Device</sam
1296 l:NameIdentifier>
1297         </saml:Subject>
1298         <saml:Attribute AttributeName="role"
1299 AttributeNamespace="http://nemo.intertrust.com/2004/attribute">
1300
1301             <saml:AttributeValue>ObjectProviderClient</saml:AttributeValue>
1302         </saml:Attribute>
1303     </saml:AttributeStatement>
1304     <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
1305         <ds:SignedInfo>
1306             <ds:CanonicalizationMethod
1307 Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
1308                 <ec:InclusiveNamespaces/>
1309             </ds:CanonicalizationMethod>
1310
1311             <ds:SignatureMethod
1312 Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>

```

```

1313         <ds:Reference URI="#ID1117583305016">
1314             <ds:Transforms>
1315                 <ds:Transform
1316 Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
1317                 <ds:Transform
1318 Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
1319                     <ec:InclusiveNamespaces/>
1320                 </ds:Transform>
1321             </ds:Transforms>
1322             <ds:DigestMethod
1323 Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1324
1325             <ds:DigestValue>0oKaxQ/MPAZ/bjtNT2I/EiIfVi8=</ds:DigestValue>
1326         </ds:Reference>
1327     </ds:SignedInfo>
1328     <ds:SignatureValue>Fb9C...ytM=</ds:SignatureValue>
1329     <ds:KeyInfo>
1330         <ds:X509Data>
1331
1332         <ds:X509Certificate>MIIC...cwE=</ds:X509Certificate>
1333         </ds:X509Data>
1334     </ds:KeyInfo>
1335 </ds:Signature>
1336 </saml:Assertion>
1337 </wsse:Security>
1338 </S11:Header>
1339
1340 <!-- SOAP Body -->
1341 <S11:Body
1342     xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
1343 wss-wssecurity-utility-1.0.xsd"
1344     wsu:Id="NEMO_ID14">
1345
1346     <!-- Payload -->
1347     <ObjectRequestPayload xmlns="urn:services:provide-objects:schema">
1348     </ObjectRequestPayload>
1349 </S11:Body>
1350 </S11:Envelope>

```

### 1351 3.1.6.1.2 Response Message

```

1352
1353 <?xml version="1.0" encoding="UTF-8"?>
1354 <S11:Envelope
1355     xmlns:S11="http://schemas.xmlsoap.org/soap/envelope/">
1356     <S11:Header>
1357         <wsse:Security
1358             xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-
1359 200401-wss-wssecurity-secext-1.0.xsd"
1360             xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
1361 wss-wssecurity-utility-1.0.xsd"
1362             xmlns:enc="http://www.w3.org/2001/04/xmlenc#"
1363             xmlns:ds="http://www.w3.org/2000/09/xmldsig#"

```

```

1364     xmlns:nemosec=
1365         "http://nemo.intertrust.com/2005/10/security"
1366     xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#"
1367     S11:mustUnderstand="1">
1368
1369     <!-- Protocol declaration -->
1370     <nemosec:ProtocolDeclaration
1371         URI="http://nemo.intertrust.com/2005/10/security/secure-
1372 protocol/basic/1.0"
1373         wsu:Id="NEMO_ID4"
1374
1375         nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
1376 protocol">
1377         <nemosec:Step Type="response"/>
1378         <nemosec:Reference
1379
1380             nemosec:TargetUsage="http://nemo.intertrust.com/2005/10/security/secu
1381 re-protocol/basic/1.0#response-encryptedMessageKey"
1382             URI="#NEMO_ID1"/>
1383         <nemosec:Reference
1384
1385             nemosec:TargetUsage="http://nemo.intertrust.com/2005/10/security/secu
1386 re-protocol/basic/1.0#response-signature"
1387             URI="#NEMO_ID3"/>
1388         </nemosec:ProtocolDeclaration>
1389
1390     <!-- Timestamp -->
1391     <wsu:Timestamp
1392         wsu:Id="NEMO_ID6"
1393
1394         nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
1395 protocol/basic/1.0#response-timestamp">
1396         <wsu:Created>2005-06-15T02:08:21.95</wsu:Created>
1397         <wsu:Expires>2005-06-15T03:08:21.95</wsu:Expires>
1398     </wsu:Timestamp>
1399
1400     <!-- Nonce -->
1401     <wsse:Nonce
1402         wsu:Id="NEMO_ID7"
1403
1404         nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
1405 protocol/basic/1.0#response-nonce"
1406         >gFyj9hobPcY=</wsse:Nonce>
1407
1408     <!-- Returned nonce -->
1409     <wsse:Nonce
1410         wsu:Id="NEMO_ID8"
1411
1412         nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
1413 protocol/basic/1.0#response-returnedNonce"
1414         >UU4o0CriFXA=</wsse:Nonce>
1415

```

```

1416     <!-- ToNode -->
1417     <nemosec:ToNode
1418         wsu:Id="NEMO_ID9"
1419
1420         nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
1421 protocol/basic/1.0#response-toNode"
1422         >urn:nemo:node:Device</nemosec:ToNode>
1423
1424     <!-- FromNode -->
1425     <nemosec:FromNode
1426         wsu:Id="NEMO_ID10"
1427
1428         nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
1429 protocol/basic/1.0#response-fromNode"
1430
1431         >1.3.6.1.4.1.7584.1.1.1=urn:nemo:node:ObjectProvider</nemosec:FromNode
1432 e>
1433
1434     <!-- Encrypted message key -->
1435     <enc:EncryptedKey
1436         Id="NEMO_ID1">
1437         <enc:EncryptionMethod
1438             Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-
1439 mgflp"/>
1440         <enc:CipherData>
1441             <enc:CipherValue>qXLi...Pr4=</enc:CipherValue>
1442         </enc:CipherData>
1443         <enc:ReferenceList>
1444             <enc>DataReference URI="#NEMO_ID15"/>
1445             <enc>DataReference URI="#NEMO_ID16"/>
1446         </enc:ReferenceList>
1447     </enc:EncryptedKey>
1448
1449     <!-- Self-encrypted message key -->
1450     <wsse:BinarySecurityToken
1451
1452         ValueType="http://nemo.intertrust.com/2005/10/security/BST/SymmetricK
1453 ey"
1454         EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
1455 wss-soap-message-security-1.0#Base64Binary"
1456         wsu:Id="NEMO_ID11"
1457
1458         nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
1459 protocol/basic/1.0#response-messageKey"
1460         >r40Md5PL7psz5V4KLNg8oQ==</wsse:BinarySecurityToken>
1461
1462     <!-- Message signing certificate chain -->
1463     <wsse:BinarySecurityToken
1464         ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-
1465 200401-wss-x509-token-profile-1.0#X509PKIPathv1"
1466         EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
1467 wss-soap-message-security-1.0#Base64Binary"

```

```

1468         wsu:Id="NEMO_ID2"
1469
1470         nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
1471 protocol/basic/1.0#response-signingKey"
1472         >MIIC...xUM=</wsse:BinarySecurityToken>
1473
1474         <!-- Signature -->
1475         <ds:Signature
1476             Id="NEMO_ID3">
1477             <ds:SignedInfo>
1478                 <ds:CanonicalizationMethod
1479 Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
1480                     <ec:InclusiveNamespaces/>
1481                 </ds:CanonicalizationMethod>
1482
1483                 <ds:SignatureMethod
1484 Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
1485
1486                 <!-- Protocol declaration -->
1487                 <ds:Reference URI="#NEMO_ID4">
1488                     <ds:Transforms>
1489                         <ds:Transform
1490 Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
1491                             <ec:InclusiveNamespaces/>
1492                         </ds:Transform>
1493                     </ds:Transforms>
1494                     <ds:DigestMethod
1495 Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1496
1497                     <ds:DigestValue>47xQ5MHQOobKhmitz26dSE0IlXw=</ds:DigestValue>
1498                     </ds:Reference>
1499
1500                     <ds:DigestValue>c84XiUsfpEk9YUHeMFtyd7l/wsw=</ds:DigestValue>
1501                     </ds:Reference>
1502
1503                 <!-- Timestamp -->
1504                 <ds:Reference URI="#NEMO_ID6">
1505                     <ds:Transforms>
1506                         <ds:Transform
1507 Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
1508                             <ec:InclusiveNamespaces/>
1509                         </ds:Transform>
1510                     </ds:Transforms>
1511                     <ds:DigestMethod
1512 Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1513
1514                     <ds:DigestValue>RLx5nlpdPLdoVckxj9TgH/7mZY0=</ds:DigestValue>
1515                     </ds:Reference>
1516
1517                 <!-- Nonce -->
1518                 <ds:Reference URI="#NEMO_ID7">
1519                     <ds:Transforms>

```

```

1520         <ds:Transform
1521 Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
1522         <ec:InclusiveNamespaces/>
1523         </ds:Transform>
1524     </ds:Transforms>
1525     <ds:DigestMethod
1526 Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1527
1528     <ds:DigestValue>I3IeanHd42nMhBt7QdDNWvPOE0w=</ds:DigestValue>
1529     </ds:Reference>
1530
1531     <!-- Returned nonce -->
1532     <ds:Reference URI="#NEMO_ID8">
1533         <ds:Transforms>
1534             <ds:Transform
1535 Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
1536             <ec:InclusiveNamespaces/>
1537             </ds:Transform>
1538         </ds:Transforms>
1539         <ds:DigestMethod
1540 Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1541
1542         <ds:DigestValue>EJU0huKed7KSAQWnz4MdXZQHZmI=</ds:DigestValue>
1543         </ds:Reference>
1544
1545         <!-- ToNode -->
1546         <ds:Reference URI="#NEMO_ID9">
1547             <ds:Transforms>
1548                 <ds:Transform
1549 Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
1550                 <ec:InclusiveNamespaces/>
1551                 </ds:Transform>
1552             </ds:Transforms>
1553             <ds:DigestMethod
1554 Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1555
1556             <ds:DigestValue>mLKsYekEKNRiydR/JW//nyrlJhc=</ds:DigestValue>
1557             </ds:Reference>
1558
1559             <!-- FromNode -->
1560             <ds:Reference URI="#NEMO_ID10">
1561                 <ds:Transforms>
1562                     <ds:Transform
1563 Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
1564                     <ec:InclusiveNamespaces/>
1565                     </ds:Transform>
1566                 </ds:Transforms>
1567                 <ds:DigestMethod
1568 Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1569
1570                 <ds:DigestValue>3W7L7XXuqEr9azm7wpgksZ3TZjI=</ds:DigestValue>
1571                 </ds:Reference>

```

```

1572
1573     <!-- Message key -->
1574     <ds:Reference URI="#NEMO_ID11">
1575         <ds:Transforms>
1576             <ds:Transform
1577 Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
1578                 <ec:InclusiveNamespaces/>
1579             </ds:Transform>
1580         </ds:Transforms>
1581         <ds:DigestMethod
1582 Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1583
1584         <ds:DigestValue>jaLOXfkP07TOaU+LoOD/m3x7DNo=</ds:DigestValue>
1585     </ds:Reference>
1586
1587     <!-- SOAP Body -->
1588     <ds:Reference URI="#NEMO_ID14">
1589         <ds:Transforms>
1590             <ds:Transform
1591 Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
1592                 <ec:InclusiveNamespaces/>
1593             </ds:Transform>
1594         </ds:Transforms>
1595         <ds:DigestMethod
1596 Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1597
1598         <ds:DigestValue>L/wM/Au+zdkRCy007IIBLJC84qM=</ds:DigestValue>
1599     </ds:Reference>
1600 </ds:SignedInfo>
1601 <ds:SignatureValue>U/rj...R1Q=</ds:SignatureValue>
1602 <ds:KeyInfo>
1603     <wsse:SecurityTokenReference>
1604         <!-- Message signing certificate chain -->
1605         <wsse:Reference URI="#NEMO_ID2"/>
1606     </wsse:SecurityTokenReference>
1607 </ds:KeyInfo>
1608 </ds:Signature>
1609 </wsse:Security>
1610 </S11:Header>
1611
1612 <!-- SOAP Body -->
1613 <S11:Body
1614     xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
1615 wss-wssecurity-utility-1.0.xsd"
1616     wsu:Id="NEMO_ID14">
1617
1618     <!-- Message payload -->
1619     <ObjectResponsePayload
1620         xmlns="urn:services:provide-objects:schema">
1621     </ObjectResponsePayload>
1622 </S11:Body>
1623 </S11:Envelope>

```

### 3.1.6.1.3 Confirmation Message

```
1624
1625
1626 <?xml version="1.0" encoding="UTF-8"?>
1627 <S11:Envelope
1628     xmlns:S11="http://schemas.xmlsoap.org/soap/envelope/">
1629     <S11:Header>
1630         <wsse:Security
1631             xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-
1632 200401-wss-wssecurity-secext-1.0.xsd"
1633             xmlns:wss="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
1634 wss-wssecurity-utility-1.0.xsd"
1635             xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
1636             xmlns:enc="http://www.w3.org/2001/04/xmlenc#"
1637             xmlns:nemosec=
1638                 "http://nemo.intertrust.com/2005/10/security"
1639             xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#"
1640             S11:mustUnderstand="1">
1641
1642         <!-- Protocol declaration -->
1643         <nemosec:ProtocolDeclaration
1644             URI="http://nemo.intertrust.com/2005/10/security/secure-
1645 protocol/basic/1.0"
1646
1647             nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
1648 protocol">
1649             <nemosec:Step Type="confirmation"/>
1650             <nemosec:Reference
1651
1652                 nemosec:TargetUsage="http://nemo.intertrust.com/2005/10/security/secu
1653 re-protocol/basic/1.0#confirmation-encryptedMessageKey"
1654                 URI="#NEMO_ID1"/>
1655             <nemosec:Reference
1656
1657                 nemosec:TargetUsage="http://nemo.intertrust.com/2005/10/security/secu
1658 re-protocol/basic/1.0#confirmation-signature"
1659                 URI="#NEMO_ID3"/>
1660             </nemosec:ProtocolDeclaration>
1661
1662         <!-- Timestamp -->
1663         <wsu:Timestamp
1664             wsu:Id="NEMO_ID4"
1665
1666             nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
1667 protocol/basic/1.0#confirmation-timestamp">
1668             <wsu:Created>2005-06-15T02:08:22.2</wsu:Created>
1669             <wsu:Expires>2005-06-15T03:08:22.2</wsu:Expires>
1670         </wsu:Timestamp>
1671
1672         <!-- Returned nonce -->
1673         <wsse:Nonce
1674             wsu:Id="NEMO_ID5"
```

```

1675
1676     nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
1677 protocol/basic/1.0#confirmation-returnedNonce"
1678         >gFyj9hobPcY=</wsse:Nonce>
1679
1680     <!-- ToNode -->
1681     <nemosec:ToNode
1682         wsu:Id="NEMO_ID6"
1683
1684     nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
1685 protocol/basic/1.0#confirmation-toNode"
1686         >urn:nemo:node:ObjectProvider</nemosec:ToNode>
1687
1688     <!-- FromNode -->
1689     <nemosec:FromNode
1690         wsu:Id="NEMO_ID7"
1691
1692     nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
1693 protocol/basic/1.0#confirmation-fromNode"
1694
1695     >1.3.6.1.4.1.7584.1.1.1=urn:nemo:node:Device</nemosec:FromNode>
1696
1697     <!-- Encrypted message key -->
1698     <enc:EncryptedKey
1699         Id="NEMO_ID1">
1700         <enc:EncryptionMethod
1701             Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-
1702 mgflp"/>
1703         <enc:CipherData>
1704             <enc:CipherValue>DJ38...4yM=</enc:CipherValue>
1705         </enc:CipherData>
1706         <enc:ReferenceList>
1707             <enc>DataReference URI="#NEMO_ID10"/>
1708             <enc>DataReference URI="#NEMO_ID11"/>
1709         </enc:ReferenceList>
1710     </enc:EncryptedKey>
1711
1712     <!-- Self-encrypted message key -->
1713     <wsse:BinarySecurityToken
1714
1715     ValueType="http://nemo.intertrust.com/2005/10/security/BST/SymmetricK
1716 ey"
1717     EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
1718 wss-soap-message-security-1.0#Base64Binary"
1719         wsu:Id="NEMO_ID8"
1720
1721     nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
1722 protocol/basic/1.0#confirmation-messageKey"
1723         >/wuLG7KSi3HoHGak3Ibw7Q==</wsse:BinarySecurityToken>
1724
1725     <!-- Message signing key certificate chain -->
1726     <wsse:BinarySecurityToken

```

```

1727         ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-
1728 200401-wss-x509-token-profile-1.0#X509PKIPathv1"
1729         EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
1730 wss-soap-message-security-1.0#Base64Binary"
1731         wsu:Id="NEMO_ID2"
1732
1733         nemosec:Usage="http://nemo.intertrust.com/2005/10/security/secure-
1734 protocol/basic/1.0#confirmation-signingKey"
1735         >MIIC...xUM=</wsse:BinarySecurityToken>
1736
1737         <!-- Signature -->
1738         <ds:Signature
1739             Id="NEMO_ID3">
1740             <ds:SignedInfo>
1741                 <ds:CanonicalizationMethod
1742 Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
1743                 <ec:InclusiveNamespaces/>
1744                 </ds:CanonicalizationMethod>
1745
1746                 <ds:SignatureMethod
1747 Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
1748
1749                 <!-- Timestamp -->
1750                 <ds:Reference URI="#NEMO_ID4">
1751                     <ds:Transforms>
1752                         <ds:Transform
1753 Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
1754                         <ec:InclusiveNamespaces/>
1755                         </ds:Transform>
1756                     </ds:Transforms>
1757                     <ds:DigestMethod
1758 Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1759                     <ds:DigestValue>CsZ4+lyUFTaZaN95saqjrYv4RRc=</ds:DigestValue>
1760                     </ds:Reference>
1761
1762                 <!-- Nonce -->
1763                 <ds:Reference URI="#NEMO_ID5">
1764                     <ds:Transforms>
1765                         <ds:Transform
1766 Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
1767                         <ec:InclusiveNamespaces/>
1768                         </ds:Transform>
1769                     </ds:Transforms>
1770                     <ds:DigestMethod
1771 Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1772                     <ds:DigestValue>qo/2ozEhHKdXezzk41jlpwFF/4=</ds:DigestValue>
1773                     </ds:Reference>
1774
1775                 <!-- ToNode -->
1776                 <ds:Reference URI="#NEMO_ID6">

```

```

1779         <ds:Transforms>
1780             <ds:Transform
1781 Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
1782                 <ec:InclusiveNamespaces/>
1783             </ds:Transform>
1784         </ds:Transforms>
1785         <ds:DigestMethod
1786 Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1787
1788         <ds:DigestValue>6WL57mr3KLwGZW6KhmC62sNfz9I=</ds:DigestValue>
1789         </ds:Reference>
1790
1791         <!-- FromNode -->
1792         <ds:Reference URI="#NEMO_ID7">
1793             <ds:Transforms>
1794                 <ds:Transform
1795 Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
1796                     <ec:InclusiveNamespaces/>
1797                 </ds:Transform>
1798             </ds:Transforms>
1799             <ds:DigestMethod
1800 Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1801
1802             <ds:DigestValue>AWr11XUMv7jzEoy4JAuQwWQ6X0g=</ds:DigestValue>
1803             </ds:Reference>
1804
1805             <!-- Self-encrypted message key -->
1806             <ds:Reference URI="#NEMO_ID8">
1807                 <ds:Transforms>
1808                     <ds:Transform
1809 Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
1810                         <ec:InclusiveNamespaces/>
1811                     </ds:Transform>
1812                 </ds:Transforms>
1813                 <ds:DigestMethod
1814 Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1815
1816                 <ds:DigestValue>ov+JClUnR6LBChe59TfDoQ0xJT4=</ds:DigestValue>
1817                 </ds:Reference>
1818
1819                 <!-- SOAP Body -->
1820                 <ds:Reference URI="#NEMO_ID9">
1821                     <ds:Transforms>
1822                         <ds:Transform
1823 Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
1824                             <ec:InclusiveNamespaces/>
1825                         </ds:Transform>
1826                     </ds:Transforms>
1827                     <ds:DigestMethod
1828 Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1829
1830                     <ds:DigestValue>TFg5gdgBYzBBk1k3jQBD5l6jxBw=</ds:DigestValue>

```

```
1831     </ds:Reference>
1832   </ds:SignedInfo>
1833   <ds:SignatureValue>6umm...ijk=</ds:SignatureValue>
1834   <ds:KeyInfo>
1835     <wsse:SecurityTokenReference>
1836       <!-- Message signing certificate chain -->
1837       <wsse:Reference URI="#NEMO_ID2"/>
1838     </wsse:SecurityTokenReference>
1839   </ds:KeyInfo>
1840 </ds:Signature>
1841 </wsse:Security>
1842 </S11:Header>
1843
1844 <!-- SOAP Body -->
1845 <S11:Body
1846   wsu:Id="NEMO_ID9">
1847   <ObjectConfirmPayload xmlns="urn:services:provide-
1848 objects:schema"/>
1849 </S11:Body>
1850 </S11:Envelope>
```

## 1851 **3.2 Security Considerations**

1852 Message senders should be aware of the remote chance of a vulnerability to an element  
1853 substitution attack. WS-Security [\[WS-SEC\]](#) recommends the use of bare-name (shortcut)  
1854 XPointers to reference XML document elements. Using this mechanism, there is no way to tell  
1855 from a signature alone if a referenced element was originally located within a message header,  
1856 within the Security header, within the message body, or in some other location. However, only  
1857 unusual message circumstances will create a vulnerability to such an attack.

## 1858 **4 NEMO Trust Management Bindings**

### 1859 **4.1 Introduction (Informative)**

1860 This section describes bindings of NEMO trust management mechanisms—in particular, the use  
1861 of SAML-specified URIs for NEMO node identifiers, the use of X.509 certificates for NEMO  
1862 node authentication, the use of SAML attribute assertions with NEMO nodes, and the definition  
1863 of a special NEMO node “role” attribute.

1864 It is intended that NEMO roles be used in authorization decisions, both by service consumers and  
1865 by service providers. If a role is asserted by an unknown authority, it may be necessary to  
1866 authorize the role authority on the basis of the role authority’s own role. This implies a recursive  
1867 chaining algorithm for verifying roles, based on role assertion authorization policy. Unlike other  
1868 uses of authorization roles, NEMO roles are not organized in a privilege hierarchy, nor can they  
1869 be quickly assumed and shed.

1870 Also described is a means of simultaneously authenticating and authorizing a NEMO node via  
1871 TLS server authentication. This binding is intended for use by small portable devices. This  
1872 mechanism allows one-sided authentication and authorization using a TLS facility that is  
1873 probably already present on the small device. However, use of this mechanism requires that  
1874 certificate authorities issue certificates with authorization semantics. Also, a TLS client cannot  
1875 signal to a TLS server the client’s preferred certificate trust anchor, so servers must offer only a  
1876 single certificate chain, and clients must be provisioned to trust a sufficient set of certificate  
1877 authorities.

### 1878 **4.2 URI NEMO Identifier Binding**

1879 NEMO nodes SHALL be identified by a canonicalized URI reference of length no more than  
1880 1024 characters. URI references are specified in [\[RFC2396\]](#). The URI MUST be an absolute  
1881 identifier (as opposed to a URI relative to a base identifier). The scheme of the URI MUST  
1882 support a canonicalization algorithm, so that there is a unique canonicalized URI character  
1883 sequence defined for the node identifier.

#### 1884 **4.2.1 Canonical Representation of HTTP URLs**

1885 The canonical URI character sequence corresponding to an HTTP URL [\[RFC2616\]](#) SHALL  
1886 observe the following properties.

- 1887 • The URL SHALL NOT contain the default port for that URI-reference;
- 1888 • Host names MUST be in lower case only;
- 1889 • Scheme names MUST be in lower case only;
- 1890 • The abs\_path component MUST NOT be empty;
- 1891 • Characters in the “reserved” and “unsafe” sets (see [\[RFC2396\]](#)) SHALL be given their  
1892 " "% " HEX HEX" encoding;
- 1893 • Characters other than those in the “reserved” and “unsafe” sets (see [\[RFC2396\]](#)) SHALL  
1894 NOT be given their " "% " HEX HEX" encoding;

- 1895       • Any "% HEX HEX" encodings SHALL use lower case representations of the  
1896       hexadecimal digits.

## 1897   **4.2.2 Canonical Representation of URNs**

1898   The canonical URI character sequence corresponding to a URN [\[RFC2141\]](#) SHALL observe the  
1899   following properties.

- 1900       • The leading "urn:" token SHALL be in lower case;
- 1901       • The Namespace ID SHALL be in lower case;
- 1902       • Characters in the "reserved" and "unsafe" sets (see [\[RFC2396\]](#)) SHALL be given their  
1903       "% HEX HEX" encoding;
- 1904       • Characters other than those in the "reserved" and "unsafe" sets (see [\[RFC2396\]](#)) SHALL  
1905       NOT be given their "% HEX HEX" encoding;
- 1906       • Any "% HEX HEX" encodings SHALL use lowercase representations of the  
1907       hexadecimal digits.
- 1908       • The Namespace Specific String must be canonicalized according to any conventions  
1909       defined by the Namespace ID.

## 1910   **4.3 X.509 Authentication Binding**

### 1911   **4.3.1 X.500 Object Identifiers**

1912   Intertrust's private enterprise object identifier arc is given here.

```
1913       id-itru OBJECT IDENTIFIER ::= {iso(1) identified-organization(3)  
1914       dod(6) internet(1) private(4) enterprise(1) intertrust(7584)}
```

1915   This arc can be referenced as <http://oid.elibel.tm.fr/1.3.6.1.4.1.7584> and is repeated here for your  
1916   convenience.

#### 1917   **4.3.1.1 NEMO**

1918   A sub-arc of the Intertrust object identifier arc is devoted to NEMO.

```
1919       id-nemo OBJECT IDENTIFIER ::= {id-itru nemo(1)}
```

#### 1920   **4.3.1.2 Name Attributes**

1921   The following arc is used within NEMO to identify X.500 name attributes, for use within X.500  
1922   distinguished names.

```
1923       id-nemo-nat OBJECT IDENTIFIER ::= {id-nemo nameAttribute(1)}
```

#### 1924   **4.3.1.3 Extended Key Usages**

1925   The following arc is used within NEMO to identify X.509 extended key usages within X.509  
1926   certificates.

```
1927       id-nemo-xku OBJECT IDENTIFIER ::= {id-nemo extendedKeyUsage(2)}
```

## 1928 **4.3.2 X.509 Certificate Use**

1929 The public keys held by NEMO nodes SHALL be bound to the node identifiers via the use of a  
1930 public key infrastructure, using X.509 certificate validation, as specified in [\[PKIX\]](#). If the  
1931 subject of an X.509 certificate is a NEMO node, the `subject` field SHALL contain a  
1932 distinguished name, as specified in §4.3.4, and the `subjectAltName` extension MUST NOT  
1933 be marked critical.

1934 The subject key identifier extension of an X.509 certificate whose subject name is a NEMO node  
1935 ID SHALL be a 160-bit SHA-1 hash of the `subjectPublicKey` field, this being the first  
1936 method suggested by [\[RFC3280\]](#) section 4.2.1.2.

1937 The X.509 extended key usage extension MAY be present and may be marked critical. NEMO  
1938 nodes are not required to process the X.509 fields `issuerUniqueID` and  
1939 `subjectUniqueID`.

## 1940 **4.3.3 Signature Algorithms**

1941 NEMO nodes MUST support the following certificate signature algorithms [\[PKIXALGS\]](#).

- 1942 • `sha1WithRSAEncryption`

## 1943 **4.3.4 Distinguished Names**

1944 NEMO node distinguished names MUST contain exactly one distinguished name attribute,  
1945 which SHALL be a “uri” attribute encoded as a UTF8String. The UTF-8 character sequence of  
1946 the description distinguished name attribute MUST be an encoding of the canonical URI  
1947 character sequence of the NEMO node identifier URI.

1948 The `uri` attribute is defined using ASN.1 syntax and conventions in [\[X.520\]](#) as

```
1949     uri ATTRIBUTE ::= {  
1950         WITH SYNTAX UTF8String {ub-uri}  
1951         EQUALITY MATCHING RULE caseExactMatch  
1952         SUBSTRINGS MATCHING RULE caseExactSubstringsMatch  
1953         ID id-nat-uri }  
1954  
1955     id-nat-uri OBJECT IDENTIFIER ::= {id-nemo-nat 1}  
1956     ub-uri INTEGER ::= 1024
```

## 1957 **4.3.5 Certificate Revocation**

1958 A NEMO node SHOULD NOT rely on or use any certificate identified in a valid certificate  
1959 revocation list (CRL), as specified in [\[RFC3280\]](#).

## 1960 **4.3.6 Certificate and Key Renewal**

1961 NEMO nodes capable of secure time MUST NOT use or accept expired certificates for the  
1962 purpose of authenticating NEMO nodes. A NEMO node MUST NOT use or accept for the  
1963 purpose of authenticating a NEMO node a certificate whose expiration date is after the issue date  
1964 of a valid CRL received from the certificate authority that issued the certificate.

1965 A certificate authority (CA) MUST establish a rollover period before the CA’s certificate signing  
1966 key expires. During the rollover period the CA MUST issue certificates signed with the new

1967 certificate signing key, as well as a pair of rollover certificates certifying the expiring key with  
1968 the new key, and certifying the new key with the expiring key. The rollover certificates MUST  
1969 expire at the same time as the CA's older certificate signing key.

## 1970 **4.4 SAML NEMO Node Attribute Assertions**

1971 A NEMO node may be associated with trusted attributes. Attribute associations and values may  
1972 be asserted by the issuance of `Assertion` elements containing  
1973 `<saml:AttributeStatement>` elements, as specified in [\[SAML1.1\]](#). SAML assertions  
1974 MUST be signed. SAML assertions SHALL NOT inherit signatures from non-SAML elements  
1975 ([\[SAML1.1\]](#), Section 5.3). If the `<saml:Subject>` element of a NEMO attribute assertion  
1976 contains a `<saml:NameIdentifier>` element, the `Format` attribute of the  
1977 `<saml:NameIdentifier>` element MUST be present, and MUST have the following value.

```
1978 &nemo;/saml/name-format/uri
```

1979 This format identifier indicates that the subject name identifier is a URI (see [RFC2396](#)). Also,  
1980 the `Issuer` attribute of the `<saml:Assertion>` element MUST contain a URI. If the  
1981 subject or issuer of a SAML assertion is a NEMO node, the content of the  
1982 `<saml:NameIdentifier>` element or `Issuer` attribute SHALL be the URI identifier of the  
1983 NEMO node.

1984 Note (Informative): NEMO attributes are made trustworthy by the presence of trusted assertions.  
1985 NEMO attributes may change with time, but NEMO attributes should not be variables that  
1986 change faster than trusted assertions can be distributed.

### 1987 **4.4.1 Processing Rules for SAML Attributes**

#### 1988 **4.4.1.1 Attribute Association**

1989 The processing model of SAML attribute statements is not completely specified in [\[SAML1.1\]](#).  
1990 In the present NEMO binding, a SAML attribute statement signifies that the subject is associated  
1991 with the attribute and with at least the indicated attribute values. Note the following:

- 1992 1. In the present binding, two SAML attribute statements appearing in the same SAML  
1993 assertion with the same subject and attribute designator are functionally equivalent to a  
1994 single attribute statement with the combined set of attribute values.
- 1995 2. Some policy processing rules may employ “negation as failure,” meaning that a policy  
1996 succeeds if an attribute is not “known” by the processor (through the presence of  
1997 attribute assertions in a message or in a processor cache) to be associated with a specific  
1998 value. Under such processing rules, an attribute assertion that associates an attribute  
1999 with two values is not functionally equivalent to two attribute assertions that separately  
2000 associate the attribute with the same two values, since in the latter case a single assertion  
2001 may be presented alone to the assertion processor, and the other withheld. For example, a  
2002 policy might require an attribute to be associated with exactly one of the values x or y.  
2003 The policy will succeed if only one of the assertions (x or y) is presented, but the policy  
2004 will fail if only a combined assertion (x and y) is available.
- 2005 3. In the present binding, there is no means, using only SAML attribute statements and  
2006 without further specification, to assert that a subject is not associated with a given  
2007 attribute value.

2008 4. The specification of an attribute type may include further processing rules, such as  
2009 restrictions on the number of associated values.

#### 2010 **4.4.1.2 SAML Conditions**

2011 NEMO processors **MUST** process `NotBefore` and `NotOnOrAfter` attributes in a  
2012 `<saml:Conditions>` element. Support for other SAML conditions is **OPTIONAL**.  
2013 Processors **MUST** reject an assertion that contains unsupported conditions.

#### 2014 **4.4.1.3 Validity Caching**

2015 Before relying on a SAML assertion, a NEMO processor **SHOULD** authenticate the issuer of the  
2016 assertion. A NEMO processor **MAY** apply additional conditions and procedures before  
2017 validating the reliability of a SAML assertion. For example, a processor may require that the  
2018 issuer have a certain fixed identity, or else that the issuer be associated with certain attributes or  
2019 attribute values.

2020 Once a NEMO processor has validated a SAML assertion that does not contain the  
2021 `<saml:DoNotCacheCondition>` element, the processor **MAY** continue to rely upon the  
2022 assertion as long as the conditions of the assertion are satisfied, without re-validating the  
2023 assertion at each instance of reliance. A NEMO processor that supports the  
2024 `<saml:DoNotCacheCondition>` element **MUST** re-validate a SAML assertion containing  
2025 a `<saml:DoNotCacheCondition>` element at each instance of reliance upon the SAML  
2026 assertion.

2027 Note: A SAML assertion containing the `<saml:DoNotCacheCondition>` element cannot  
2028 be relied upon after the issuer's signing key has expired.

#### 2029 **4.4.2 NEMO Role Attributes**

2030 This binding defines a NEMO node attribute called "roles". The NEMO roles attribute **SHALL**  
2031 be identified as a SAML attribute with namespace

2032 `&nemo;/attribute`

2033 and name

2034 `role`

2035 A NEMO node **MAY** be associated with zero or more roles. NEMO role attributes **SHALL** have  
2036 associated values whose type is `xs:anyURI`. The `Issuer` and `Subject` of a NEMO role  
2037 attribute statement **MUST** be NEMO nodes identified by their NEMO node identifiers.

#### 2038 **4.4.3 NEMO Attribute Reliance Policy (Informative)**

2039 If the issuer of an attribute assertion is a NEMO node, other NEMO nodes may govern their  
2040 reliance on the attribute assertion according to the roles held by the attribute assertion issuer.

#### 2041 **4.5 SSL/TLS Service Authorization**

2042 Transport Layer Security [\[TLS\]](#) is a protocol for point-to-point message security. A NEMO  
2043 client may simultaneously authenticate and authorize a service provider node by successfully  
2044 establishing a TLS session with server-side authentication. Message integrity and confidentiality

2045 can be provided by the TLS transport, so services SHOULD NOT require higher-level message  
2046 integrity and confidentiality protection when using TLS service authorization.

2047 A service MUST indicate its support for TLS service authorization in its WSDL service  
2048 description document. A service MUST also indicate in its service description document whether  
2049 TLS client authentication is required (§5). NEMO clients using TLS service authorization are not  
2050 required to support TLS client authentication.

2051 Certificate authorities that issue certificates for the purpose of NEMO TLS service authorization  
2052 SHALL issue certificates that comply with the X.509 profile specified by [\[WAPCertProf\]](#). End  
2053 entity certificates whose subjects are NEMO nodes SHALL include Subject fields containing  
2054 the NEMO node's identifier, formatted as specified in §4.3.4.

2055 **Notes** (informative):

2056 1. TLS doesn't provide a way for a client to specify the certificate authorities trusted for  
2057 server authentication. (TLS does provide a way for a server to specify the certificate  
2058 authorities trusted for client authentication.) Servers supporting TLS authorization  
2059 typically will be issued only a single certificate to be offered to clients. Clients must be  
2060 configured with sufficient trust anchor certificates to authenticate all desired services.

2061 2. TLS specifies authentication via X.509 certificates. While X.509 public key  
2062 infrastructures are normally used only for binding certificate holders' names to their  
2063 public keys, NEMO clients using TLS authorization may rely on certificate authorities to  
2064 establish trust and authorization of service nodes.

2065 3. NEMO does not specify the certification practices of certification authorities, or how  
2066 clients manage certification trust anchors for TLS service authorization. In particular,  
2067 NEMO does not specify whether some or all of the TLS service authorization trust  
2068 anchors within a client can be managed or manipulated by device holders, or whether  
2069 trust anchors are securely held away from device holders (tamper resistant).

## 2070 **4.5.1 Certificate Revocation**

2071 A NEMO client using TLS service authorization SHOULD NOT rely on or use any certificate  
2072 identified in a valid certificate revocation list (CRL), as specified in [\[RFC3280\]](#).

## 2073 **4.5.2 Certificate and Key Renewal**

2074 NEMO nodes capable of secure time that support TLS service authorization MUST NOT use or  
2075 accept expired certificates in TLS sessions. A NEMO node using TLS service authorization  
2076 MUST NOT use or accept a certificate whose expiration date is after the issue date of a valid  
2077 CRL received from the certificate authority that issued the certificate.

2078 A certificate authority (CA) MUST establish a rollover period before the CA's certificate signing  
2079 key expires. During the rollover period, the CA MUST issue certificates signed with the new  
2080 certificate signing key, as well as a pair of rollover certificates certifying the expiring key with  
2081 the new key, and certifying the new key with the expiring key. The rollover certificates MUST  
2082 expire at the same time as the CA's older certificate signing key.

2083

## 4.6 Annex

2084 This annex includes all of the ASN.1 type and value definitions contained in this specification  
2085 (§4) in the form of the ASN.1 module NEMO.

```
2086 NEMO {iso(1) identified-organization(3) dod(6) internet(1) private(4)  
2087 enterprise(1) intertrust(7584) nemo(1)}
```

```
2088 DEFINITIONS ::=
```

```
2089 BEGIN
```

```
2090
```

```
2091 -- EXPORTS All --
```

```
2092
```

```
2093 IMPORTS
```

```
2094 -- from Intertrust X.500 Object Identifier Specification
```

```
2095     id-itrn
```

```
2096         FROM Intertrust {iso(1) identified-organization(3) dod(6)
```

```
2097             internet(1) private(4) enterprise(1) intertrust(7584)}
```

```
2098
```

```
2099 id-nemo OBJECT IDENTIFIER ::= {id-itrn nemo(1)}
```

```
2100 id-nat OBJECT IDENTIFIER ::= {id-nemo nameAttribute(1)}
```

```
2101 id-xku OBJECT IDENTIFIER ::= {id-nemo extendedKeyUsage(2)}
```

```
2102
```

```
2103 uri ATTRIBUTE ::= {
```

```
2104     WITH SYNTAX UTF8String {ub-uri}
```

```
2105     EQUALITY MATCHING RULE caseExactMatch
```

```
2106     SUBSTRINGS MATCHING RULE caseExactSubstringsMatch
```

```
2107     ID id-nat-uri }
```

```
2108
```

```
2109 id-nat-uri OBJECT IDENTIFIER ::= {id-nat uri(1)}
```

```
2110 ub-uri INTEGER ::= 1024
```

```
2111
```

```
2112 END -- NEMO
```

```
2113
```

## 2114 **5 NEMO Policy Bindings**

### 2115 **5.1 Overview**

2116 This section specifies bindings that can be used to express policies defining the security  
2117 requirements for the NEMO Secure Messaging Protocol bindings. This section assumes a  
2118 working knowledge of the following specifications:

- 2119 NEMO Security Bindings (§3)
- 2120 NEMO Trust Management Bindings (§4)
- 2121 WS-Security [\[WS-SEC\]](#)
- 2122 WS-SecureConversation [\[WS-SCON\]](#)
- 2123 WS-Trust [\[WS-Trust\]](#)
- 2124 WS-Policy [\[WS-POL\]](#)
- 2125 WS-SecurityPolicy [\[WS-SEC-POL\]](#)
- 2126 WS-PolicyAttachment [\[WS-POL-ATTCH\]](#)
- 2127 Web Services Description Language 1.1 [\[WSDL 1.1\]](#)
- 2128 Security Assertion Markup Language [\[SAML1.1\]](#)

### 2129 **5.2 NEMO Web Service Policy Binding**

2130 The NEMO Web Service Policy Binding defines a way for NEMO services to advertise their  
2131 policies within a WSDL service description document [\[WSDL 1.1\]](#). For a given service, these  
2132 policies define the particular protocol options followed to establish a secure message channel, as  
2133 well as the service's authorization requirements and guarantees.

2134 Policies SHOULD make reference to the protocols observed by a service, and MUST NOT  
2135 conflict with any choices specified in the protocol definition. Any such conflicting policy  
2136 elements SHOULD be ignored and MUST NOT be enforced. If a service description excludes  
2137 policy specifications for options available in the protocol binding specification, or if an assertion  
2138 is explicitly declared optional, then any permitted option MAY be used.

2139 For example, a service's policy implementing a Web Service binding of the Basic Secure  
2140 Messaging Protocol defined in (§3) may specify whether confidentiality is required or rejected,  
2141 and must not override the encryption algorithm choices defined by the binding. If the service's  
2142 policy does not specify confidentiality protection policy, then confidentiality protection is  
2143 optional, per the Basic Secure Messaging Protocol.

2144 Policies MAY use policy assertions other than those defined in this binding to express policies,  
2145 and MAY refine policy scope, so long as the policies do not conflict with the protocol definition.

2146 WS-PolicyAttachment [\[WS-POL-ATTCH\]](#) may be used to provide scope for policies. Unless  
2147 policies are explicitly attached to particular scopes, they are assumed to be in-scope for all  
2148 communication where they are applicable.

2149 The usages defined in [\[WS-POL\]](#) SHALL be used inside the policy assertions to declare protocol  
2150 option selections.

## 2151 **5.2.1 Protocol Policy**

2152 A protocol may be referenced by its identifier. The protocol is refined using policies, which can  
2153 be attached to the protocol using WS-PolicyAttachment [\[WS-POL-ATTCH\]](#).

### 2154 **5.2.1.1 Security Tokens**

2155 A `<wssp:SecurityToken>` element may assert the presence of a token named in the  
2156 protocol. The attribute `nemosec:Usage` defined in §3 SHALL be used to signal the usage of  
2157 the token to which the `<wssp:SecurityToken>` element corresponds.

### 2158 **5.2.1.2 nemop:Usage Attribute**

2159 The `nemop:Usage` attribute MAY be included within policy assertions to indicate that the  
2160 assertion has a distinguished role in a certain context, such as a communications protocol.  
2161 NEMO message processors can use the `nemop:Usage` attribute as a hint to locate policy  
2162 assertions relating to specific contexts.

### 2163 **5.2.1.3 nemosec:Usage Attribute**

2164 The `nemosec:Usage` attribute MAY be used within a `<wssp:SecurityToken>` element  
2165 to identify a policy assertion that indicates requirements or capabilities related to a token with a  
2166 distinguished role in a certain context, such as a communications protocol. NEMO message  
2167 processors can use the `nemosec:Usage` attribute within a `<wssp:SecurityToken>`  
2168 element as a hint to locate message tokens relating to specific contexts.

### 2169 **5.2.1.4 MessageAge**

2170 Whenever timestamp support is defined as optional by the protocol, the policy MAY specify  
2171 whether the timestamp is used in the particular interaction, and MAY specify what maximum  
2172 message age is accepted. A `<wssp:MessageAge>` assertion SHALL be used to signal  
2173 message timestamp usage.

2174 A policy assertion specifying the use of an optional timestamp within the following protocols  
2175 (§3) MAY contain the attribute `nemop:Usage` with the value indicated below.  
2176

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

### 2177 **5.2.1.5 Nonce**

2178 Whenever nonce usage is defined as optional by the protocol, the policy MAY specify whether a  
2179 nonce is used in the the particular interaction. The element `<nemop:Nonce>` SHALL be used  
2180 to signal whether a nonce is used by the service.

2181 The syntax for the <nemop:Nonce> element is  
 2182 .../Nonce  
 2183 This element is a policy assertion expressing the requirement for a nonce in a message.  
 2184 .../{any}  
 2185 This is an extensibility mechanism allowing other elements describing the nonce.  
 2186 .../@{any}  
 2187 This is an extensibility mechanism allowing attributes describing the nonce.  
 2188 A policy assertion specifying the use of an optional nonce within the following protocols (§3)  
 2189 MAY contain the attribute nemop:Usage with the value indicated below.  
 2190

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

2191 **5.2.1.6 Signed Message Key**

2192 Whenever a protocol defines as optional that the secret key used to encrypt the message contents  
 2193 be signed, the policy MAY specify whether the signed message key is required to be present in  
 2194 the message. It is RECOMMENDED that this assertion also be explicitly specified in the policy  
 2195 when the protocol requires the secret key to be signed conditioned on both integrity and  
 2196 confidentiality being present. A <wssp:SecurityToken> element with TokenType of  
 2197 &nemosec;/SymmetricKey SHALL be used to signal whether the symmetric key is  
 2198 required (see §5.2.1.1).

2199 **5.2.1.7 Signed Challenge / Signed Challenge Response**

2200 Whenever a signed challenge/response usage is defined as optional by the protocol, the policy  
 2201 MAY specify whether signed challenge/response is used in the particular interaction. The  
 2202 elements <nemop:SignChallenge> and <nemop:SignChallengeResponse> SHALL  
 2203 be used to signal whether signed challenge and response are used by the service.

2204 The syntax for the <nemop:SignChallenge> element is

2205 .../SignChallenge

2206 This element is a policy assertion expressing the requirement for a signed challenge in a  
 2207 message.

2208 .../{any}

2209 This is an extensibility mechanism allowing other elements describing the signed challenge.

2210 .../@{any}

2211 This is an extensibility mechanism allowing attributes describing the signed challenge.

2212 The syntax for the `<nemop:SignChallengeResponse>` element is  
 2213 `.../SignChallengeResponse`  
 2214 This element is a policy assertion expressing the requirement for a signed challenge response  
 2215 in a message.  
 2216 `.../{any}`  
 2217 This is an extensibility mechanism allowing other elements describing the signed challenge  
 2218 response.  
 2219 `.../@{any}`  
 2220 This is an extensibility mechanism allowing attributes describing the signed challenge  
 2221 response.  
 2222 A policy assertion specifying the use of an optional signed challenge within the following  
 2223 protocols (§3) MAY contain the attribute `nemop:Usage` with the value indicated below.  
 2224

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

### 2225 **5.2.1.8 Message Integrity**

2226 Whenever message integrity usage is defined as optional by the protocol, the policy MAY  
 2227 specify whether message integrity is used in the the particular interaction. The element  
 2228 `<wssp:Integrity>` SHALL be used to signal whether message integrity is used by the  
 2229 service.  
 2230 A policy assertion specifying the optional use of integrity protection within the following  
 2231 protocols (§3) MAY contain the attribute `nemop:Usage` with the value indicated below.  
 2232

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

### 2233 **5.2.1.9 Message Confidentiality**

2234 Whenever message confidentiality usage is defined as optional by the protocol, the policy MAY  
 2235 specify whether message confidentiality is used in the particular interaction. A  
 2236 `<wssp:Confidentiality>` element SHALL be used to signal whether message  
 2237 confidentiality is used by the service.  
 2238 A policy assertion specifying the optional use of confidentiality protection within the following  
 2239 protocols (§3) MAY contain the attribute `nemop:Usage` with the value indicated below.

2240

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

### 2241 **5.2.1.10 Message Parts**

2242 This specification defines a dialect of the `wssp:MessageParts` mechanism defined in [\[WS-](#)  
2243 [SEC-POL\]](#). This dialect extends, and thereby includes all the message part functions defined in,  
2244 the dialect specified in [\[WS-POL-ASSRT\]](#), identified by the URI

2245 `http://schemas.xmlsoap.org/2002/12/wsse#part`

2246 `/wssp:MessageParts/@Dialect`

2247 When using the dialect specified here, the `@Dialect` attribute should be the URI

2248 `http://nemo.intertrust.com/2004/policy#part`

2249 `/wssp:MessageParts`

2250 When using the mechanism specified here, the contents of the `<wssp:MessageParts>`  
2251 element is a string of the form

2252 `nemop:Token(usage)`

2253 where “usage” represents a URI indicating the usage of the message element in the  
2254 messaging protocol.

2255 Note: The attribute `nemosec:Usage` defined in §3 MAY be attached to message elements as a  
2256 hint to receiving processors trying to locate message elements with a given protocol usage.  
2257 Receiving processors are nonetheless ultimately responsible for locating and verifying the usage  
2258 of elements that have usages within the semantics of a defined protocol.

### 2259 **5.2.1.11 XML Element Encryption Policy**

2260 According to [\[WS-SEC\]](#), the tags of an `<S11:Body>` element SHALL NOT be encrypted. A  
2261 policy requiring the encryption of the `<S11:Body>` element indicates one of three methods for  
2262 encrypting the SOAP Body.

- 2263 • A processor MAY encrypt the `<S11:Body>` element according to [\[XMLENC\]](#), by  
2264 replacing the contents of the `<S11:Body>` element with an `<enc:EncryptedData>`  
2265 element with a `Type` attribute having the value  
2266 `http://www.w3.org/2001/04/xmlenc#Content`

2267 This is the RECOMMENDED method of encrypting an `<S11:Body>` element that has  
2268 content. This method MAY be applied even if the unencrypted `<S11:Body>` element  
2269 has no content. In this case, the `<enc:EncryptedData>` element’s cipher data  
2270 SHALL resolve to an empty octet sequence.

- 2271 • If the unencrypted `<S11:Body>` element contains only whitespace and a single  
2272 child element, then a processor MAY satisfy the encryption policy by encrypting the

2273 child element, as specified in [\[XMLENC\]](#). In this case, the entirety of the child element  
2274 must be encrypted, including the child element's tags.

2275 • If the unencrypted <S11:Body> element has no content, then a processor MAY satisfy  
2276 the encryption policy by not altering the empty <S11:Body> element. This is the  
2277 RECOMMENDED method of encrypting an <S11:Body> element that has no content.

2278 According to [\[WS-SEC\]](#), the <S11:Header> and <S11:Envelope> elements of a SOAP  
2279 message SHALL NOT be encrypted. To satisfy policy requiring the encryption of a message  
2280 element that is not an <S11:Body>, <S11:Header> or <S11:Envelope> element, a  
2281 processor SHALL encrypt the entire element, including the element tags, as specified in  
2282 [\[XMLENC\]](#).

## 2283 **5.2.2 Message Security Policy**

2284 The Message Security Policy governs message-level security between a client and a service. In  
2285 particular, message security policy MAY require that a particular message protocol definition be  
2286 observed.

2287 Message Security policy MAY be attached to the WSDL components, as described in Section 4  
2288 of the WS-PolicyAttachment specification [\[WS-POL-ATTCH\]](#).

### 2289 **5.2.2.1 Protocol Assertion**

2290 A policy MAY make reference to the protocol that an operation implements, or the protocol step  
2291 that a message represents. A <nemop:ProtocolAssertion> element SHALL be used to  
2292 specify a particular protocol or protocol step.

2293 The syntax for <nemop:ProtocolAssertion> is as follows:

```
2294 <ProtocolAssertion wsu:Id="..."?>  
2295   <Reference URI="..."/?>  
2296   <nemosec:Step index="..." type="..."/?>  
2297 </ProtocolAssertion>
```

2299 The following describes the elements defined above.

2300 /ProtocolAssertion

2301 This contains the protocol assertion.

2302 /ProtocolAssertion/Reference

2303 This element contains a URI identifying the protocol definition that is asserted by this policy  
2304 assertion.

2305 /ProtocolAssertion/Reference/@URI

2306 This attribute specifies a URI identifying the protocol definition.

2307 /ProtocolAssertion/nemosec:Step

2308 This element MAY be used when the ProtocolAssertion is attached to a message. The  
2309 <nemosec:Step> element indicates the step in a protocol that the message represents.  
2310 See §3 for a specification of the children of this element.

2311 /ProtocolAssertion/@{any}

2312 This is an extensibility mechanism to allow additional attributes, based on schemas, to be  
2313 added.

2314 /ProtocolAssertion/{any}

2315 This is an extensibility mechanism to allow different (extensible) types of security  
2316 information, based on a schema, to be passed.

### 2317 **5.2.2.2 Profile Assertion**

2318 The Profile policy assertion applies to operations. The assertion references a NEMO profile that  
2319 the service operation implements. A <nemosec:Profile> element defined in §3 SHALL be  
2320 used within a <wsp:Policy> element to assert conformance to a particular NEMO profile.

## 2321 **5.2.3 Application Security Policy**

2322 In addition to the Message Security Policy, the service MAY publish its application policies. The  
2323 policies can be described using the same elements as defined in §5.2.2, as well as other policy  
2324 elements. In particular, SAML [\[SAML1.1\]](#) Security Token assertions may be used to express  
2325 application policy assertions.

2326 Application Security policy MAY be attached to the WSDL components, as described in Section  
2327 4 of the WS-PolicyAttachment specification [\[WS-POL-ATTCH\]](#).

### 2328 **5.2.3.1 SAML Attribute Assertion Token**

2329 Whenever a SAML attribute assertion token [\[SAML1.1\]](#) is used by the service, the service policy  
2330 MAY signal attributes and attribute values associated with the requestor NEMO node. This  
2331 SHALL be expressed using a <SecurityToken> element declared in the scope of the  
2332 service's policy with a TokenType of wssp:SAMLAssertion, as specified in WS-  
2333 SecurityPolicy [\[WS-SEC-POL\]](#).

2334 When the wssp:SecurityToken/wssp:TokenType is wssp:SAMLAssertion, the  
2335 wssp:SecurityToken/wssp:Claims element MAY contain a set of  
2336 <nemop:NameValuePairDescription> elements. The <wssp:SecurityToken>  
2337 policy assertion describes the valid assertion and presentation of SAML attribute statement(s).  
2338 SAML attributes' names and values SHALL be described via the  
2339 <nemop:NameValuePairDescription> element. The <wssp:SecurityToken>  
2340 element's asserted policy is the association of the subject NEMO node with the indicated  
2341 attributes and values, as described in NEMO Trust Management Bindings (§4).

2342 If the SecurityToken/Claims element contains more than one child element, then the  
2343 effect of the SecurityToken policy assertion is the same as the effect of a <wsp:All>  
2344 element containing several SecurityToken policy assertions, each with one of the child  
2345 elements within the SecurityToken/Claims element.

2346 /wssp:SecurityToken/wssp:Claims/NameValuePairDescription

2347 This element identifies a node attribute and optionally values to be associated with the  
2348 subject NEMO node.

2349 /wssp:SecurityToken/wssp:Claims/NameValuePairDescription/@Name

2350 The name of the NEMO node attribute.

2351 /wssp:SecurityToken/wssp:Claims/NameValuePairDescription/@Namespace

2352 The namespace of the NEMO node attribute.

2353 /wssp:SecurityToken/wssp:Claims/NameValuePairDescription/ValuePattern

2354 This element contains a value pattern. Each ValuePattern in a  
 2355 NameValuePairDescription must match an attribute value associated with the subject  
 2356 NEMO node and the indicated attribute.

2357 Note: It is possible for two distinct <nemop:ValuePattern> elements to be satisfied by a  
 2358 single value, if the <nemop:ValuePattern> elements signal overlapping patterns.

2359 /wssp:SecurityToken/wssp:Claims/NameValuePairDescription/ValuePattern/@MatchType

2360 This optional string attribute determines how to match a NEMO node attribute value with the  
 2361 contents of the <nemop:ValuePattern> element. This binding defines the value

2362 `wssp:Exact`

2363 to signal that the <nemop:ValuePattern> element matches only its contents, subject to  
 2364 a canonicalization algorithm.

2365 The default value for this attribute is wssp:Exact.

2366 /wssp:SecurityToken/wssp:Claims/NameValuePairDescription/ValuePattern/@Canonicalization

2367 This optional URI attribute signals the canonicalization algorithm used to match XML data  
 2368 to the contents of the ValuePattern. The default canonicalization algorithm for element  
 2369 content is XML Exclusive Canonicalization [\[EXC-C14N\]](#). The default canonicalization  
 2370 algorithm for character data content conforms to the text node serialization specified in  
 2371 [\[EXC-C14N\]](#).

2372 The processing rules specified by the NEMO Trust Management Bindings (§4) SHALL apply  
 2373 when evaluating SAML assertions with regard to policies. Consequently, the following claims  
 2374 inside a SAML Security Token assertion are semantically equivalent:

2375 1. Multiple <nemop:NameValuePairDescription> elements with the same  
 2376 attribute name.

2377

```

2378 <wssp:Claims xmlns:example="http://www.example.com/attribute">
2379   <nemop:NameValuePairDescription
2380     Name="attributeName"
2381     Namespace="http://www.example.com/attribute">
2382     <nemop:ValuePattern>
2383       <example:AAA/>
2384     </nemop:ValuePattern>
2385   </nemop:NameValuePairDescription>
2386
2387   <nemop:NameValuePairDescription
2388     Name="attributeName"
2389     Namespace="http://www.example.com/attribute">
2390     <nemop:ValuePattern>
2391       <example:BBB/>
  
```

```
2392     </nemop:ValuePattern>
2393     </nemop:NameValuePairDescription>
2394 </wssp:Claims>
```

2395 2. A single `<nemop:NameValuePairDescription>` element enclosing multiple  
2396 `<nemop:ValuePattern>` elements

```
2397
2398 <wssp:Claims xmlns:example="http://www.example.com/attribute">
2399     <nemop:NameValuePairDescription
2400         Name="attributeName"
2401         Namespace="http://www.example.com/attribute">
2402         <nemop:ValuePattern>
2403             <example:AAA/>
2404         </nemop:ValuePattern>
2405         <nemop:ValuePattern>
2406             <example:BBB/>
2407         </nemop:ValuePattern>
2408     </nemop:NameValuePairDescription>
2409 </wssp:Claims>
```

## 2410 5.2.4 Session Policy

2411 Whenever the service supports the Secure Conversation Binding specified in §3, the Session  
2412 Service MAY publish policies for establishing the session. Messaging requirements SHALL be  
2413 specified as described in §5.2.2. Additionally, the following policies MAY be expressed.

### 2414 5.2.4.1 Session Duration

2415 Session policy SHOULD specify the session's minimum and maximum lifetime. This SHALL be  
2416 expressed using the `<nemop:SessionDuration>` element declared in the scope of the  
2417 Session Service of the service node.

2418 The syntax for this element is as follows:

```
2419
2420 <nemop:SessionDuration>
2421     <nemop:MinimumDuration>
2422         ...
2423     </nemop:MinimumDuration>
2424
2425     <nemop:MaximumDuration>
2426         ...
2427     </nemop:MaximumDuration>
2428 </nemop:SessionDuration>
```

2429 The following describes the attributes and elements listed in the schema overview above:

2430 /SessionDuration

2431 This is the policy element specifying the range of session durations for the sessions that the  
2432 Session Service can generate.

2433 /SessionDuration/MiniumDuration

2434 This is the element specifying the minimum session duration. This element is of the same  
2435 schema type as the <wssp:MessageAge> element [\[WS-SEC-POL\]](#).

2436 /SessionDuration/MaximumDuration

2437 This is the element specifying the maximum session duration. This element is of the same  
2438 schema type as the <wssp:MessageAge> element [\[WS-SEC-POL\]](#).

2439 /SessionDuration/@{any}

2440 This is an extensibility mechanism to allow additional attributes, based on schemas, to be  
2441 added.

2442 /SessionDuration/{any}

2443 This is an extensibility mechanism to allow different (extensible) ways of specifying session  
2444 duration policy, based on a schema, to be passed. Unrecognized elements SHOULD be  
2445 ignored.

## 2446 **5.2.5 Caching Policy**

2447 Policies for services that support sessions MAY include policies for caching the validity of  
2448 credentials that the client node has previously provided to the service node within the current  
2449 session. If a credential validity is cached, the client node does not need to resubmit the credential  
2450 within the session. (There may be cases when credential validity expires prematurely—for  
2451 example, if a credential is revoked.)

2452 If the client resubmits a credential whose validity has been previously cached within the current  
2453 session, the service MAY revalidate the credential and re-cache it, or the service MAY ignore the  
2454 supplied credential and keep the current cache state.

2455 The service node MAY choose to cache credential validity for longer than the session. This  
2456 specification doesn't provide a mechanism to signal that behavior.

### 2457 **5.2.5.1 Signaling Credential Caching**

2458 Policy MAY signal whether the service node caches the validity of credentials within sessions.  
2459 The policy SHALL be expressed using the nemop:Cache attribute, whose values may be  
2460 "true" or "false". If the nemop:Cache attribute is missing, its default value is "false". If a  
2461 needed credential is not cached by the service node, clients SHOULD supply the credential with  
2462 each request.

2463 Syntax:

```
2464 <...AnyCredentialPolicy nemop:Cache="..." />
```

## 2466 **5.2.6 Examples**

### 2467 **5.2.6.1 Basic Protocol – No Sessions**

```
2468 <?xml version="1.0" encoding="UTF-8"?>
```

```
2469 <definitions name="LicenseManager"
```

```
2470 targetNamespace="http://example.com/myservice"
```

```
2471
```

```

2472 xmlns:tns="http://example.com/myservice "
2473 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
2474 xmlns="http://schemas.xmlsoap.org/wsdl/"
2475 xmlns:wsp="http://schemas.xmlsoap.org/ws/2003/12/policy"
2476 xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
2477 wsswssecurity-secext-1.0.xsd"
2478 xmlns:wssp="http://schemas.xmlsoap.org/ws/2002/12/secext/"
2479 xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
2480 xmlns:nemop="http://nemo.intertrust.com/2004/policy"
2481 xmlns:nemosec="http://nemo.intertrust.com/2005/10/security">
2482 <wsp:UsingPolicy wsdl:Required="true" />
2483
2484 <!-- Service Node's Public Encryption Key -->
2485 <wsse:SecurityTokenReference
2486     nemosec:Usage="&nemosec;/secure-protocol/basic/1.0#request-
2487 encryptionKey">
2488     <wsse:Embedded>
2489         <wsse:BinarySecurityToken
2490             ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-
2491 200401-wss-x509-token-profile-1.0#X509PKIPathv1"
2492             EncodingType="wsse:Base64Binary">
2493                 ...X509 Certificate...
2494             </wsse:BinarySecurityToken>
2495         </wsse:Embedded>
2496     </wsse:SecurityTokenReference>
2497
2498 <!-- Service Node's Public Signing Key -->
2499 <wsse:SecurityTokenReference
2500     nemosec:Usage="&nemosec;/secure-protocol/basic/1.0#response-
2501 signingKey">
2502     <wsse:Embedded>
2503         <wsse:BinarySecurityToken
2504             ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-
2505 200401-wss-x509-token-profile-1.0#X509PKIPathv1"
2506             EncodingType="wsse:Base64Binary">
2507                 ...X509 Certificate...
2508             </wsse:BinarySecurityToken>
2509         </wsse:Embedded>
2510     </wsse:SecurityTokenReference>
2511
2512 <!--Service's role assertion -->
2513 <wsse:SecurityTokenReference
2514     nemosec:Usage="&nemo;/attribute/role">
2515     <wsse:Embedded>
2516         <saml:Assertion
2517             AssertionID="..."
2518             IssueInstant="..."
2519             Issuer="urn:nemo:node:CA"
2520             MajorVersion="1"
2521             MinorVersion="1">
2522             <saml:AttributeStatement>
2523                 <saml:Subject>

```

```

2524         <saml:NameIdentifier>
2525             uri=urn:nemo:node:A
2526         </saml:NameIdentifier>
2527     </saml:Subject>
2528     <saml:Attribute
2529         AttributeName="role"
2530         AttributeNamespace=".../attribute">
2531         <saml:AttributeValue>
2532             MyServiceRole
2533         </saml:AttributeValue>
2534     </saml:Attribute>
2535 </saml:AttributeStatement>
2536 <ds:Signature>
2537 </ds:Signature>
2538 </saml:Assertion>
2539 </wsse:Embedded>
2540 </wsse:SecurityTokenReference>
2541
2542 <types>
2543     <xsd:schema>
2544         <xsd:complexType name="RequestType" />
2545         <xsd:complexType name="ResponseType" />
2546         <xsd:element name="RequestPayload" type="RequestType" />
2547         <xsd:element name="ResponsePayload"
2548             type="ResponseType" />
2549     </xsd:schema>
2550 </types>
2551
2552 <message name="Request">
2553     <part name="Request" element="RequestPayload" />
2554 </message>
2555
2556 <message name="Response">
2557     <part name="Response" element="ResponsePayload" />
2558 </message>
2559
2560 <portType name="MyPortType">
2561     <operation name="MyOperation">
2562         <input name="Request" message="Request" />
2563         <output name="Response" message="Response" />
2564     </operation>
2565 </portType>
2566
2567 <binding name="MyPortTypeSoapBinding" type="MyPortType">
2568     <wsdlsoap:binding style="document" transport="...soap/http" />
2569     <operation name="MyOperation">
2570         <wsp:PolicyReference URI="#MyOperationPolicy" />
2571     <wsdlsoap:operation/>
2572     <input>
2573         <wsp:PolicyReference URI="#RequestPolicy" />
2574         <wsdlsoap:body use="literal" namespace="..." />
2575     </input>

```

```

2576     <output>
2577         <wsp:PolicyReference URI="#ResponsePolicy"/>
2578         <wsdlsoap:body use="literal" namespace="..." />
2579     </output>
2580 </operation>
2581 </binding>
2582
2583 <service name="MyService">
2584     <port name="MyPort" binding="MyPortTypeSoapBinding">
2585         <wsdlsoap:Address location="http://..." />
2586     </port>
2587 </service>
2588
2589 <!--MyOperation Policy -->
2590 <wsp:Policy wsu:Id="MyOperationPolicy">
2591     <!--Profile -->
2592     <nemosec:Profile URI="&nemo;/profile/main"/>
2593     <!--Protocol -->
2594     <nemop:ProtocolAssertion>
2595         <nemop:Reference URI="&nemosec;/secure-
2596             protocol/basic/1.0"/>
2597     </nemop:ProtocolAssertion>
2598 </wsp:Policy>
2599
2600 <!--Request Policy -->
2601 <wsp:Policy wsu:Id="RequestPolicy">
2602
2603     <!--Protocol -->
2604     <nemop:ProtocolAssertion>
2605         <nemop:Reference URI="&nemosec;/secure-
2606             protocol/basic/1.0"/>
2607         <nemosec:Step type="request"/>
2608     </nemop:ProtocolAssertion>
2609
2610     <!--Client's encryption key -->
2611     <wssp:SecurityToken
2612         nemosec:Usage="&nemosec;/secure-
2613             protocol/basic/1.0#response-encryptionKey"/>
2614
2615     <nemop:Nonce
2616         nemop:Usage="&nemosec;/secure-
2617             protocol/basic/1.0/policyAssertion#nonce"/>
2618     <wssp:MessageAge Age="3600"
2619         nemop:Usage="&nemosec;/secure-
2620             protocol/basic/1.0/policyAssertion#timestamp"/>
2621
2622     <!--client's signature key -->
2623     <wssp:SecurityToken
2624         nemosec:Usage="&nemosec;/secure-
2625             protocol/basic/1.0#request-signingKey">
2626         <wsse:TokenType>http://docs.oasis-open.org/wss/2004/01/oasis-
2627         200401-wss-x509-token-profile-1.0#X509PKIPathv1</wsse:TokenType>

```

```

2628     <wssp:TokenIssuer>
2629         ...Trusted Roots...
2630     </wssp:TokenIssuer>
2631 </wssp:SecurityToken>
2632
2633 <!-- The keys to be used for confidentiality and integrity -->
2634 <!-- are specified by the protocol binding -->
2635 <wssp:Confidentiality
2636     nemop:Usage="&nemosec;/secure-
2637     protocol/basic/1.0/policyAssertion#confidentiality">
2638     <wssp:MessageParts Dialect="&nemop;#part">
2639         nemop:Token(&nemosec;/secure-
2640             protocol/basic/1.0#request-messageKey)
2641         wsp:Body( )
2642     </wssp:MessageParts>
2643 </wssp:Confidentiality>
2644 <wssp:Integrity
2645     nemop:Usage="&nemosec;/secure-
2646     protocol/basic/1.0/policyAssertion#integrity">
2647     <wssp:MessageParts Dialect="&nemop;#part">
2648         nemop:Token(&nemosec;/secure-
2649             protocol/basic/1.0#request-messageKey)
2650         wsp:Body( )
2651         nemop:Token(&nemosec;/.../request-timestamp)
2652         nemop:Token(&nemosec;/.../request-nonce)
2653         nemop:Token(&nemosec;/.../request-toNode)
2654     </wssp:MessageParts>
2655 </wssp:Integrity>
2656
2657 <!-- client's Role (application policy attribute token) -->
2658 <wsp:ExactlyOne>
2659     <wssp:SecurityToken>
2660         <wssp:TokenType>wsse:SAMLAssertion</wssp:TokenType>
2661         <wssp:TokenIssuer>...Trusted Roots...
2662     </wssp:TokenIssuer>
2663     <wssp:Claims>
2664         <nemop:NameValuePairDescription
2665             Name="role"
2666             Namespace="&nemop;/attribute">
2667             <nemop:ValuePattern>
2668                 MyClientRole
2669             </nemop:ValuePattern>
2670         </nemop:NameValuePairDescription>
2671     </wssp:Claims>
2672 </wssp:SecurityToken>
2673
2674 <wssp:SecurityToken>
2675     <wssp:TokenType>wsse:SAMLAssertion</wssp:TokenType>
2676     <wssp:TokenIssuer>...Trusted Roots...
2677 </wssp:TokenIssuer>
2678     <wssp:Claims>
2679         <nemop:NameValuePairDescription

```

```

2680         Name="role"
2681         Namespace="&nemopi;/attribute">
2682         <nemopi:ValuePattern>
2683             MyAlternateClientRole
2684         </nemopi:ValuePattern>
2685     </nemopi:NameValuePairDescription>
2686 </wssp:Claims>
2687 </wssp:SecurityToken>
2688 </wsp:ExactlyOne>
2689 </wsp:Policy>
2690
2691 <!--Response Policy -->
2692 <wsp:Policy wsu:Id="ResponsePolicy">
2693
2694     <!--Protocol -->
2695     <nemopi:ProtocolAssertion>
2696         <nemopi:Reference URI="&nemosec;/secure-
2697             protocol/basic/1.0"/>
2698         <nemosec:Step type="response"/>
2699     </nemopi:ProtocolAssertion>
2700
2701     <!--client's encryption key -->
2702     <wssp:SecurityToken
2703         nemosec:Usage="&nemosec;/secure-
2704             protocol/basic/1.0#response-encryptionKey">
2705         <wsse:TokenType>http://docs.oasis-open.org/wss/2004/01/oasis-
2706 200401-wss-x509-token-profile-1.0#X509PKIPathv1</wsse:TokenType>
2707         <wssp:TokenIssuer>
2708             ...Trusted Roots...
2709         </wssp:TokenIssuer>
2710     </wssp:SecurityToken>
2711
2712     <!--client's signature key -->
2713     <wssp:SecurityToken
2714         nemosec:Usage="&nemosec;/secure-
2715             protocol/basic/1.0#request-signingKey"/>
2716
2717     <nemopi:Nonce
2718         nemopi:Usage="&nemosec;/secure-
2719             protocol/basic/1.0/policyAssertion#nonce"/>
2720     <wssp:MessageAge Age="3600"
2721         nemopi:Usage="&nemosec;/secure-
2722             protocol/basic/1.0/policyAssertion#timestamp"/>
2723     <wssp:Confidentiality
2724         nemopi:Usage="&nemosec;/secure-
2725             protocol/basic/1.0/policyAssertion#confidentiality">
2726         <wssp:MessageParts Dialect="&nemopi;#part">
2727             nemopi:Token(&nemosec;/secure-
2728                 protocol/basic/1.0#response-messageKey)
2729             wsp:Body()
2730         </wssp:MessageParts>
2731     </wssp:Confidentiality>

```

```

2732     <wssp:Integrity
2733         nemop:Usage="&nemosec;/secure-
2734             protocol/basic/1.0/policyAssertion#integrity">
2735     <wssp:MessageParts Dialect="&nemop;#part">
2736         nemop:Token(&nemosec;/secure-
2737             protocol/basic/1.0#response-messageKey)
2738         wsp:Body()
2739         nemop:Token(&nemosec;/.../response-timestamp)
2740         nemop:Token(&nemosec;/.../response-nonce)
2741         nemop:Token(&nemosec;/.../response-toNode)
2742     </wssp:MessageParts>
2743 </wssp:Integrity>
2744 </wsp:Policy>
2745 ...
2746 </wsdl:definitions>

```

2747

## 2748 **5.2.6.2 Secure Conversation Protocol**

```

2749 <?xml version="1.0" encoding="UTF-8"?>
2750 <definitions name="SessionService"
2751     targetNamespace="http://example.com/sessionService "
2752     xmlns:tns="http://example.com/sessionService "
2753     xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
2754     xmlns="http://schemas.xmlsoap.org/wsdl/"
2755     xmlns:wSDL="http://schemas.xmlsoap.org/wsdl/"
2756     xmlns:wsp=
2757         "http://schemas.xmlsoap.org/ws/2003/12/policy"
2758     xmlns:wsse=
2759         "http://docs.oasis-open.org/wss/2004/01/oasis-
2760         200401-wsswssecurity-secext-1.0.xsd">
2761     xmlns:wssp=
2762         "http://schemas.xmlsoap.org/ws/2002/12/secext"
2763     xmlns:nemop="http://nemo.intertrust.com/2004/policy"
2764     xmlns:nemosec=http://nemo.intertrust.com/2005/10/security>
2765
2766     <wsp:UsingPolicy wsdL:Required="true" />
2767
2768     <!-- Service Node's Public Encryption Key -->
2769     <wsse:SecurityTokenReference
2770         nemosec:Usage="&nemosec;/secure-protocol/secure-
2771             conversation/1.0#establishment-request-
2772             encryptionKey">
2773         <wsse:Embedded>
2774             <wsse:BinarySecurityToken
2775                 ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-
2776                 200401-wss-x509-token-profile-1.0#X509PKIPathv1"
2777                 EncodingType="wsse:Base64Binary">
2778                 ...X509 Certificate...
2779             </wsse:BinarySecurityToken>
2780         </wsse:Embedded>

```

```

2782 </wsse:SecurityTokenReference>
2783
2784 <!-- Service Node's Public Signing Key -->
2785 <wsse:SecurityTokenReference
2786     nemosec:Usage="&nemosec;/secure-protocol/secure-
2787     conversation/1.0#establishment-response-signingKey">
2788     <wsse:Embedded>
2789         <wsse:BinarySecurityToken
2790             ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-
2791 200401-wss-x509-token-profile-1.0#X509PKIPathv1"
2792             EncodingType="wsse:Base64Binary">
2793                 ...X509 Certificate...
2794             </wsse:BinarySecurityToken>
2795         </wsse:Embedded>
2796     </wsse:SecurityTokenReference>
2797
2798 <!--Service's role assertion -->
2799 <wsse:SecurityTokenReference
2800     nemosec:Usage="&nemo;/attribute/role">
2801     <wsse:Embedded>
2802         <saml:Assertion
2803             AssertionID="..."
2804             IssueInstant="..."
2805             Issuer="urn:nemo:node:CA"
2806             MajorVersion="1"
2807             MinorVersion="1">
2808             <saml:AttributeStatement>
2809                 <saml:Subject>
2810                     <saml:NameIdentifier>
2811                         uri=urn:nemo:node:A
2812                     </saml:NameIdentifier>
2813                 </saml:Subject>
2814                 <saml:Attribute
2815                     AttributeName="role"
2816                     AttributeNamespace=".../attribute">
2817                     <saml:AttributeValue>
2818                         MyServiceRole
2819                     </saml:AttributeValue>
2820                 </saml:Attribute>
2821             </saml:AttributeStatement>
2822             <ds:Signature>
2823             </ds:Signature>
2824         </saml:Assertion>
2825     </wsse:Embedded>
2826 </wsse:SecurityTokenReference>
2827
2828 <types>
2829     <xsd:schema>
2830         <xsd:import namespace="&nemosec;"/>
2831         <xsd:complexType name="RequestType"/>
2832         <xsd:complexType name="ResponseType"/>
2833         <xsd:import namespace="http://.../trust"/>

```

```

2834     <xsd:import namespace=http://.../sc" />
2835     </xsd:schema>
2836 </types>
2837
2838 <message name="SessionRequest">
2839     <part name="Request" element="wst:RequestSecurityToken" />
2840 </message>
2841
2842 <message name="SessionResponse">
2843     <part name="Response"
2844         element="wst:RequestSecurityTokenResponse" />
2845 </message>
2846
2847 <message name="Request">
2848     <part name="Request" type="RequestPayloadType" />
2849 </message>
2850
2851 <message name="Response">
2852     <part name="Response" type="ResponsePayloadType" />
2853 </message>
2854
2855 <portType name="MyPortType">
2856     <operation name="MyOperation">
2857         <input name="Request" message="Request" />
2858         <output name="Response" message="Response" />
2859     </operation>
2860 </portType>
2861
2862 <portType name="SessionPortType">
2863     <operation name="EstablishSession">
2864         <input name="Request" message="SessionRequest" />
2865         <output name="Response" message="SessionResponse" />
2866     </operation>
2867 </portType>
2868
2869 <binding name="MyPortTypeSoapBinding" type="MyPortType">
2870     <wsdlsoap:binding style="document" transport="...soap/http" />
2871     <operation name="MyOperation">
2872         <wsp:PolicyReference URI="#MyOperationPolicy" />
2873         <wsdlsoap:operation/>
2874         <input>
2875             <wsp:PolicyReference URI="#RequestPolicy" />
2876             <wsdlsoap:body use="literal" namespace="..." />
2877         </input>
2878         <output>
2879             <wsp:PolicyReference URI="#ResponsePolicy" />
2880             <wsdlsoap:body use="literal" namespace="..." />
2881         </output>
2882     </operation>
2883 </binding>
2884
2885 <binding name="SessionPortTypeSoapBinding"

```

```

2886     type="SessionPortType">
2887     <wsdlsoap:binding style="document" transport="...soap/http"/>
2888     <operation name="EstablishSession">
2889         <wsp:PolicyReference URI="#EstablishSessionPolicy"/>
2890         <wsdlsoap:operation/>
2891         <input>
2892             <wsp:PolicyReference URI="#SessionRequestPolicy"/>
2893             <wsdlsoap:body use="literal" namespace="..."/>
2894         </input>
2895         <output>
2896             <wsp:PolicyReference URI="#SessionResponsePolicy"/>
2897             <wsdlsoap:body use="literal" namespace="..."/>
2898         </output>
2899     </operation>
2900 </binding>
2901
2902 <service name="MyService">
2903     <port name="MyPort" binding="MyPortTypeSoapBinding">
2904         <wsdlsoap:address location="http://..."/>
2905     </port>
2906 </service>
2907
2908 <service name="SessionService">
2909     <port name="SessionPort"
2910         binding="SessionPortTypeSoapBinding">
2911         <wsdlsoap:Address location="http://..."/>
2912     </port>
2913 </service>
2914
2915 <!--MyOperation Policy -->
2916 <wsp:Policy wsu:Id="MyOperationPolicy">
2917     <!--Profile -->
2918     <nemosec:Profile URI="&nemo;/profile/main"/>
2919     <!--Protocol -->
2920     <nemop:ProtocolAssertion>
2921         <nemop:Reference URI="&nemosec;/secure-protocol/secure-
2922             conversation/1.0"/>
2923     </nemop:ProtocolAssertion>
2924 </wsp:Policy>
2925
2926 <!-- Request Policy -->
2927 <wsp:Policy wsu:Id="RequestPolicy">
2928     <!--Protocol -->
2929     <nemop:ProtocolAssertion>
2930         <nemop:Reference URI="&nemosec;/secure-protocol/secure-
2931             conversation/1.0"/>
2932         <nemosec:Step type="request">
2933     </nemop:ProtocolAssertion>
2934     <nemop:Nonce
2935         nemop:Usage="&nemosec;/secure-
2936             protocol/secure-
2937             conversation/1.0/policyAssertion#nonce"/>

```

```

2938 <wssp:MessageAge Age="3600"
2939     nemop:Usage="&nemosec;/secure-
2940     protocol/secure-
2941     conversation/1.0/policyAssertion#timestamp"/>
2942 <wssp:Confidentiality
2943     nemop:Usage="&nemosec;/secure-
2944     protocol/secure-
2945     conversation/1.0/policyAssertion#confidentiality">
2946 <wssp:MessageParts Dialect="&nemop;#part">
2947     wsp:Body()
2948 </wssp:MessageParts>
2949 </wssp:Confidentiality>
2950 <wssp:Integrity
2951     nemop:Usage="&nemosec;/secure-
2952     protocol/secure-
2953     conversation/1.0/policyAssertion#integrity">
2954     wsp:Body()
2955     nemop:Token(&nemosec;/.../request-timestamp)
2956     nemop:Token(&nemosec;/.../request-nonce)
2957     nemop:Token(&nemosec;/.../request-toNode)
2958 </wssp:MessageParts>
2959 </wssp:Integrity>
2960 </wsp:Policy>
2961
2962 <!-- Response Policy -->
2963 <wsp:Policy wsu:Id="ResponsePolicy">
2964 <!--Protocol -->
2965 <nemop:ProtocolAssertion>
2966     <nemop:Reference URI="&nemosec;/secure-protocol/secure-
2967     conversation/1.0"/>
2968     <nemosec:Step type="response">
2969 </nemop:ProtocolAssertion>
2970 <nemop:Nonce
2971     nemop:Usage="&nemosec;/secure-
2972     protocol/secure-
2973     conversation/1.0/policyAssertion#nonce"/>
2974 <wssp:MessageAge Age="3600"
2975     nemop:Usage="&nemosec;/secure-
2976     protocol/secure-
2977     conversation/1.0/policyAssertion#timestamp"/>
2978 <wssp:Confidentiality
2979     nemop:Usage="&nemosec;/secure-
2980     protocol/secure-
2981     conversation/1.0/policyAssertion#confidentiality">
2982 <wssp:MessageParts Dialect="&nemop;#part">
2983     wsp:Body()
2984 </wssp:MessageParts>
2985 </wssp:Confidentiality>
2986 <wssp:Integrity
2987     nemop:Usage="&nemosec;/secure-
2988     protocol/secure-
2989     conversation/1.0/policyAssertion#integrity">

```

```

2990     <wssp:MessageParts Dialect="&nemop;#part">
2991         wsp:Body( )
2992         nemop:Token(&nemosec;/.../response-timestamp)
2993         nemop:Token(&nemosec;/.../response-nonce)
2994         nemop:Token(&nemosec;/.../response-toNode)
2995     </wssp:MessageParts>
2996 </wssp:Integrity>
2997 </wsp:Policy>
2998
2999 <!--Establish Session Policy -->
3000 <wsp:Policy wsu:Id="EstablishSessionPolicy">
3001     <!--Profile -->
3002     <nemosec:Profile URI="&nemo;/profile/main"/>
3003     <!--Protocol -->
3004     <nemop:ProtocolAssertion>
3005         <nemop:Reference URI="&nemosec;/secure-protocol/secure-
3006             conversation/1.0"/>
3007     </nemop:ProtocolAssertion>
3008 </wsp:Policy>
3009
3010 <!--Session Request Policy -->
3011 <wsp:Policy wsu:Id="SessionRequestPolicy">
3012     <!--Protocol -->
3013     <nemop:ProtocolAssertion>
3014         <nemop:Reference URI="&nemosec;/secure-protocol/secure-
3015             conversation/1.0"/>
3016         <nemosec:Step type="establishment-request">
3017         </nemop:ProtocolAssertion>
3018 </nemop:ProtocolAssertion>
3019
3020 <!--Service's encryption key -->
3021 <wssp:SecurityToken
3022     nemosec:Usage="&nemosec;/secure-
3023         protocol/secure-conversation/1.0#establishment-
3024         request-encryptionKey"/>
3025
3026 <!--client's signature key -->
3027 <wssp:SecurityToken
3028     nemosec:Usage="&nemosec;/secure-protocol/secure-
3029         conversation/1.0#establishment-request-signingKey">
3030     <wsse:TokenType>http://docs.oasis-open.org/wss/2004/01/oasis-
3031 200401-wss-x509-token-profile-1.0#X509PKIPathv1</wssp:TokenType>
3032     <wssp:TokenIssuer>
3033         ...Trusted Roots...
3034     </wssp:TokenIssuer>
3035 </wssp:SecurityToken>
3036
3037 <nemop:Nonce
3038     nemop:Usage="&nemosec;/secure-
3039         protocol/secure-
3040         conversation/1.0/policyAssertion#nonce"/>
3041 <wssp:MessageAge Age="3600"

```

```

3042     nemop:Usage="&nemosec;/secure-
3043     protocol/secure-
3044     conversation/1.0/policyAssertion#timestamp"/>
3045
3046 <!-- The keys to be used for confidentiality and integrity -->
3047 <!-- are specified by the protocol binding -->
3048 <wssp:Confidentiality
3049     nemop:Usage="&nemosec;/secure-
3050     protocol/secure-
3051     conversation/1.0/policyAssertion#confidentiality">
3052     <wssp:MessageParts Dialect="&nemop;#part">
3053         wsp:Body()
3054     </wssp:MessageParts>
3055 </wssp:Confidentiality>
3056 <wssp:Integrity
3057     nemop:Usage="&nemosec;/secure-
3058     protocol/secure-
3059     conversation/1.0/policyAssertion#integrity">
3060     <wssp:MessageParts Dialect="&nemop;#part">
3061         wsp:Body()
3062         nemop:Token(&nemosec;/.../establishment-request-
3063             timestamp)
3064         nemop:Token(&nemosec;/.../establishment-request-toNode)
3065     </wssp:MessageParts>
3066 </wssp:Integrity>
3067
3068 <!-- client's Role -->
3069 <wssp:ExactlyOne>
3070     <wssp:SecurityToken>
3071         <wssp:TokenType>wsse:SAMLAssertion</wssp:TokenType>
3072         <wssp:TokenIssuer>...Trusted Roots...
3073     </wssp:TokenIssuer>
3074     <wssp:Claims>
3075         <nemop:NameValuePairDescription
3076             Name="role"
3077             Namespace="&nemop;/attribute">
3078             <nemop:ValuePattern>
3079                 MyClientRole
3080             </nemop:ValuePattern>
3081         </nemop:NameValuePairDescription>
3082     </wssp:Claims>
3083 </wssp:SecurityToken>
3084
3085 <wssp:SecurityToken>
3086     <wssp:TokenType>wsse:SAMLAssertion</wssp:TokenType>
3087     <wssp:TokenIssuer>...Trusted Roots...
3088 </wssp:TokenIssuer>
3089 <wssp:Claims>
3090     <nemop:NameValuePairDescription
3091         Name="role"
3092         Namespace="&nemop;/attribute">
3093         <nemop:ValuePattern>

```

```

3094         MyAlternateClientRole
3095         </nemop:ValuePattern>
3096     </nemop:NameValuePairDescription>
3097 </wssp:Claims>
3098 </wssp:SecurityToken>
3099 </wsp:ExactlyOne>
3100 </wsp:Policy>
3101
3102 <!--Session Response Policy -->
3103 <wsp:Policy wsu:Id="SessionResponsePolicy">
3104
3105     <!--Protocol -->
3106     <nemop:ProtocolAssertion>
3107         <nemop:Reference URI="&nemosec;/secure-protocol/secure-
3108             conversation/1.0"/>
3109         <nemosec:Step type="establishment-response">
3110     </nemop:ProtocolAssertion>
3111
3112     <!--client's encryption key -->
3113     <wssp:SecurityToken
3114         nemosec:Usage="&nemosec;/secure-protocol/secure-
3115             conversation/1.0#establishment-response-encryptionKey">
3116         <wsse:TokenType>http://docs.oasis-open.org/wss/2004/01/oasis-
3117 200401-wss-x509-token-profile-1.0#X509PKIPathv1</wsse:TokenType>
3118         <wssp:TokenIssuer>
3119             ...Trusted Roots...
3120         </wssp:TokenIssuer>
3121     </wssp:SecurityToken>
3122
3123     <!--service's public signature key -->
3124     <wssp:SecurityToken
3125         nemosec:Usage="&nemosec;/secure-
3126             protocol/secure-conversation/1.0#establishment-
3127             response-signingKey"/>
3128
3129     <nemop:Nonce
3130         nemop:Usage="&nemosec;/secure-
3131             protocol/secure-
3132             conversation/1.0/policyAssertion#nonce"/>
3133     <wssp:MessageAge Age="3600"
3134         nemop:Usage="&nemosec;/secure-
3135             protocol/secure-
3136             conversation/1.0/policyAssertion#timestamp"/>
3137     <wssp:Confidentiality
3138         nemop:Usage="&nemosec;/secure-
3139             protocol/secure-
3140             conversation/1.0/policyAssertion#confidentiality">
3141         <wssp:MessageParts Dialect="&nemop;#part">
3142             wsp:Body( )
3143         </wssp:MessageParts>
3144     </wssp:Confidentiality>
3145     <wssp:Integrity

```

```
3146     nemop:Usage="&nemosec;/secure-
3147         protocol/secure-
3148         conversation/1.0/policyAssertion#integrity">
3149     <wssp:MessageParts Dialect="&nemop;#part">
3150         wsp:Body()
3151         nemop:Token(&nemosec;/.../establishment-response-
3152             timestamp)
3153         nemop:Token(&nemosec;/.../establishment-response-
3154             nonce)
3155         nemop:Token(&nemosec;/.../establishment-response-
3156             toNode)
3157     </wssp:MessageParts>
3158 </wssp:Integrity>
3159 </wsp:Policy>
3160 ...
3161 </wsdl:definitions>
```

3162

## 3163 **6 NEMO Discovery/Inspection Bindings**

### 3164 **6.1 Overview**

3165 This section specifies XML-related bindings pertaining to NEMO Inspection and Discovery.

- 3166 • Discovery – the ability to search for services offered by NEMO nodes based on different  
3167 criteria, and to obtain references to where we can bind to those services for access.
- 3168 • Inspection – given a reference to a NEMO node, the ability to query it about certain well-  
3169 defined attributes (metadata) in regards to its state, such as the descriptions of the policy  
3170 related to the services it publicly offers.

### 3171 **6.2 Service Discovery Binding**

#### 3172 **6.2.1 Overview**

3173 Discovery is the ability to search for services offered by NEMO nodes based on different criteria,  
3174 and to obtain matching references to those services. It embodies two separate aspects of Service  
3175 Oriented Architectures related to locating and inspecting networked resources:

- 3176 • Querying managed registries (databases or directory services) to locate resources
- 3177 • Dynamic, decentralized advertising and locating of (generally transient) resources.

3178 NEMO can support many different types of bindings for discovery and inspection. The  
3179 following sections describe the currently defined bindings. NEMO nodes may implement and  
3180 support more than one binding.

#### 3181 **6.2.2 WS-Discovery**

3182 This binding is based on top of the evolving WS-Discovery specifications [\[WS-Discovery\]](#).

##### 3183 **6.2.2.1 WS-Discovery Announcement**

###### 3184 **6.2.2.1.1 Description**

3185 This binding supports a mode of functionality that is intended for small unbridged subnets  
3186 including Personal Area Networks (PANs) and small unsegmented Local Area Networks  
3187 (LANs), but not segmented LAN or Wide Area Networks (WANs). The initial version of this  
3188 binding is intended for use with IP networks, although it can be adapted to other types of  
3189 networks. This binding is generally implemented on top of multicast protocols and transports  
3190 such as UDP. It also has a unicast aspect in some phases of the protocols. The following  
3191 bindings use a profile of WS-Discovery that leverages a UPnP-like type of discovery.

###### 3192 **6.2.2.1.2 Requirements (Normative)**

3193 NEMO nodes supporting this binding and offering services shall comply with WS-Discovery  
3194 [\[WS-Discovery\]](#). In particular, the following message protocols will be supported:

- 3195 • Probe – A node announcing its desire for a particular type of service by multicasting a  
3196 (Probe) message in conjunction with a node's listening for such announcements.

- 3197 • ProbeMatch – A node response with a unicast message (ProbeMatch) directed at the  
3198 source of the probe if it has matching target services.
- 3199 • Hello – A node announcing its service capabilities by multicasting a (Hello) message at  
3200 well-defined events (such as joining a network, time of day, etc.) in conjunction with a  
3201 node’s listening for such announcements.
- 3202 • Bye – A node announcing that a service capability is no longer available by multicasting  
3203 a (Bye) message at well-defined events (such as leaving a network) in conjunction with a  
3204 NEMO node’s listening for such announcements.
- 3205 • Resolve – A node announcing its desire for a particular target service based on transport-  
3206 neutral service address (ID) multicasting a (Resolve) message in conjunction with a  
3207 node’s listening for such announcements.
- 3208 • ResolveMatch – A node response with a unicast message (ResolveMatch) directed at the  
3209 source of the Resolve if it has matching target services.

3210 Multicast messages will be expected on the following communication endpoint: (Port=3702,  
3211 IPV4=239.255.255.250).

3212 As defined by the specification, Probes may support a wide variety of criteria for matching  
3213 against target services, but at a minimum nodes will support matching based on service type.

3214 As defined by the specification, ProbeMatches and ResolveMatches may describe the  
3215 communication endpoint for interacting with a node in many different ways using endpoint  
3216 references. In addition, a transport-specific communication address may also be conveyed in the  
3217 context of a <wsd:XAddrS> element.

## 3218 **6.2.2.2 WS-Discovery Discovery Proxy**

### 3219 **6.2.2.2.1 Description**

3220 This binding supports a mode of functionality that is intended for situations where announcement  
3221 (multicast) type of discovery as described above is inappropriate. This binding is appropriate  
3222 for all types of network topologies where specific communication endpoints can be addressed.

### 3223 **6.2.2.2.2 Requirements (Normative)**

3224 NEMO nodes implementing this binding shall comply with WS-Discovery, Discovery Proxy  
3225 Service [\[WS-Discovery\]](#). The actual Discovery Proxy Service interface definition is out of the  
3226 scope of the current WS-Discovery specification. We have defined a simple one that is consistent  
3227 with the intention and the behavior of the specification.

3228 This binding is generally implemented on top of unicast protocols and transports such as TCP.  
3229 The following bindings define a profile of WS-Discovery that leverages a registry-like type of  
3230 discovery.

3231 In particular, this service provides the ability to query against a registry of services, not just  
3232 against those belonging to the target node, but also against any services the target node is aware  
3233 of.

- 3234 • One round, request/response messaging pattern.

- 3235 • The input message (request) consists of a standard WS-Discovery Probe in a simple  
3236 wrapper. It will match against all services that it knows about, not just its own target  
3237 services.
- 3238 • The output message (response) consists of a bundle of zero or more WS-Discovery Probe  
3239 response(s) representing all matching target services known by the node.
- 3240 • The service is of type DiscoveryProxy, in the `http://nemo.intertrust.com/2004/discovery`  
3241 namespace.

### 3242 **6.2.2.2.3 Security**

3243 While this binding is amendable for use with the NEMO Basic Secure Messaging Protocol, one  
3244 areas that needs further work is the issue of signing QNames used within these specifications.

## 3245 **6.2.2.3 Matching Criteria**

### 3246 **6.2.2.3.1 Matching By Service Type**

3247 WS-Discovery supports matching based on service address, service type, and extensible  
3248 matching predicates called *scopes*, which are evaluated in the context of a given service.

3249 We define type-based matching based on web service port types. A service's types are defined  
3250 by one or more port types defined by a QName in a defined namespace.

### 3251 **6.2.2.3.2 Extended Forms of Matching (By Scope)**

3252 It is possible to define new types of criteria for matching with WS-Discovery. This general  
3253 facility is called *scopes*.

3254 We currently have defined some new optional types of scopes that allow for the following types  
3255 of matching:

- 3256 1. Matching by NEMO node information, such as node ID:

```
3257
3258 <wsd:Scope
3259 MatchBy="http://nemo.intertrust.com/discovery/scope/matchbynodeinfo">
3260   <xsd:element ref="nemoc:NodeInfo">
3261 </wsd:Scope>
```

3262 If this scope is supported, and if the specified node information matches for a given node, then  
3263 the scope is true, else the scope is false.

- 3264 2. Matching by support roles. The role is represented in terms of a namespace, a role value,  
3265 and an optional issuer.

```
3266
3267 <wsd:Scope
3268 MatchBy="http://nemo.intertrust.com/discovery/scope/matchbyrole">
3269   <xsd:element ref="nemodisc:RoleScopeCriteria">
3270 </wsd:Scope>
```

- 3271 3. Matching based on security policy-related tokens, such as trust anchors.

3272

```
3273 <wsd:Scope
3274 MatchBy="http://nemo.intertrust.com/discovery/scope/matchbyrole">
3275   <xsd:element ref="nemodisc:PolicyTokenScopeCriteria ">
3276   </wsd:Scope>
```

3277 If this scope is supported, and if the specified policy token information matches for a given node,  
3278 then the scope is true, else the scope is false.

3279 Please see §6.4 and §6.5 for schemas and examples related to the forms of matches possible with  
3280 this binding.

### 3281 **6.2.2.3.3 Rules for Matching**

3282 In this binding, the basic rules for matching for discovery are pretty simple. A discovery query  
3283 may consist of any of the following:

- 3284 1. Query based on service address.
- 3285 2. Query based on one or more types, where type matches are logically OR-ed .
- 3286 3. Query based on one or more scopes, where scopes must be of the **same** dialect, and  
3287 scope matches are logically OR-ed .
- 3288 4. Query 2 and Query 3 in combination logically AND-ed.

### 3289 **6.2.2.4 Service References, Properties and Parameters**

3290 If a matching service is found, and a response returned, or if a service's availability is  
3291 announced, the message will contain a description of the service. This either allows the service  
3292 to be directly interacted with or it provides enough information to bootstrap another process,  
3293 such as inspection, via the service. We call this description a service reference. The types of  
3294 service references defined in this binding are web service references.

3295 In this profile, the web service reference is embodied in terms of a WS-Addressing endpoint  
3296 reference, which will contain at least the following:

- 3297 • Service Name
- 3298 • Service Port Type(s)
- 3299 • Service Address (transport neutral)
- 3300 • Service Transport Address (Port Address)

3301 This binding also defines an optional reference parameter that may be returned to indicate the  
3302 location of a supporting service that can be used to perform inspection in regards to the matched  
3303 service in order to obtain metadata. An example of needing this parameter would be discovering  
3304 a service and needing to obtain its WSDL:

```
3305 <nemoc:InspectionReference
3306 xmlns:nemoc=http://nemo.intertrust.com/2004/core>
3307   <wsa:EndpointReference>
3308     <wsa:Address>
3309       http://localhost:9084/SimpleDiscovery/services/MetadataExchange
3310     </wsa:Address>
```

```
3312 </wsa:EndpointReference>
3313 </nemoc:InspectionReference>
```

## 3314 **6.2.3 SSDP-based Discovery**

3315 This binding is based on the Simple Service Discovery Protocol (SSDP) 1.0 specification  
3316 [\[SSDP\]](#) for discovery, and specifically relates to how NEMO nodes and their related services  
3317 may be discovered within the context of a local area network using the Simple Service Discovery  
3318 Protocol.

### 3319 **6.2.3.1 Goals**

3320 SSDP has flourished in the context of standards such as UPnP as a way of exposing and  
3321 interacting with devices and their services, particularly in the context of home environments  
3322 because of its simplicity. It is important that we leverage existing and emerging standards in this  
3323 area that balance the introduction of new capabilities with retaining compatibility with existing  
3324 UPnP environments. The goals of this discovery binding are to allow NEMO services to be  
3325 discovered via SSDP and then later inspected for more detailed information about the services.

### 3326 **6.2.3.2 Discovery Protocol Support**

3327 NEMO nodes supporting this binding will comply with the protocols for supporting service  
3328 discovery and description, as set forth in the Simple Service Discovery Protocol 1.0 [\[SSDP\]](#)  
3329 specifications.

3330 While SSDP is used as the discovery protocol in UPnP, and while some deployments may choose  
3331 to use this binding in a way that fully interoperates with UPnP deployments, there is no  
3332 obligation to do so.

3333 The current binding supports a node matching against its services based on one type of specified  
3334 search criteria conveyed in the **ST** header associated with an **ssdp:discover** request. The form of  
3335 the data conveyed in the **ST** header is a URI. NEMO is **neutral** as to the exact syntax and  
3336 semantics of the URI. Other profiles should build on top of this specification in order to create  
3337 matching schemes that mandate particular usage patterns. In supporting this binding, however,  
3338 NEMO does mandate that at a **minimum** matching based on service type, where matching is  
3339 done by exact comparison of the URI, is supported.

3340 With NEMO, it is possible to convey a variety of different search criteria in this header. Here are  
3341 some **examples** of how you could express different criteria:

3342 1. Search by Service Type.

3343 This mode of searching supports searching by service types. The currently supported  
3344 bindings for NEMO use WSDL port types to denote unique service types. The form of the  
3345 **ST** header in this case is:

```
3346 <service namespace>#<service name>
```

3347 Example:

```
3348 http://nemo.intertrust.com/services#OctopusLicenseService
```

3349 2. Search by NEMO Attribute.

3350 This mode of searching supports searching by a designated attribute, such as a role in  
3351 NEMO. The service must support the designated role. The form of the **ST** header in this  
3352 case is:

3353 **<attribute namespace>#<attribute name>#<attribute value>**

3354 Example:

3355 `http://nemo.intertrust.com/roles#role#LicenseService`

### 3356 3. Search by Trust Anchor

3357 This mode of searching supports searching by a designated trust anchor in NEMO. The  
3358 service's authentication trust anchor must be the designated trust anchor. The form of the **ST**  
3359 header in this case is:

3360 **<trust anchor distinguished name>**

3361 Example:

3362 `urn:trustanchor:1.3.6.1.4.1.7584.1.1.1=urn:ca:SystemD`

## 3363 **6.2.3.3 Service Identifier**

3364 Per the SSDP specification, there must be a way to associate an identifier with a service instance  
3365 (USN). While NEMO does not mandate the exact form of this identifier, implementers of this  
3366 binding will support returning an appropriate USN. The USN may be used in subsequent  
3367 interactions with the inspection service to identify the service instance.

3368 Example:

3369 USN: `http://localhost:9082/MarlinSettop/services/OctopusLicense`

## 3370 **6.2.3.4 Service Inspection**

3371 In order to actually communicate with a service, it may be necessary for you to go through a  
3372 service inspection interaction, for example if the USN for the discovered service is insufficient to  
3373 provide enough information to resolve and use the service. NEMO defines a binding based on  
3374 WS-MetadataExchange for inspection, where you can obtain additional information in regards to  
3375 using the service based on a unique service ID. If a node finds an appropriate match, it will send  
3376 back an SSDP discovery response, where the **Location** header contains a service endpoint  
3377 reference in the form of a URI for the inspection service to be used.

3378 Example:

3379 Location: `http://localhost:8080/service/inspection`

3380 After inspecting a service to obtain its location and other information necessary for interacting  
3381 with it, a node may cache the result to avoid going through subsequent inspection interactions.

## 3382 **6.2.3.5 Inspection Policy Identifier**

3383 The inspection service, like any other NEMO service, may be securely policy managed, and a  
3384 client must interact with the service with a well-defined NEMO security protocol.

3385 We introduce a new HTTP header (**Inspection-Policy-Id**) used in SSDP discovery responses and  
3386 in presence announcements that will designate the security policy associated with using the  
3387 inspection service. This policy may be that no security policy is in effect. The form of the policy  
3388 identifier is a URI.

3389 Example:

3390       Inspection-Policy-Id: urn:marlinpolicy:0001

## 3391 **6.2.4 Presence Announcements**

3392 In addition to NEMO nodes supporting discovery **ssdp:discover** messages, NEMO nodes may  
3393 optionally support the SSDP **ssdp:alive** and **ssdp:bye** messages for alerting interested parties to  
3394 the status of their services.

## 3395 **6.2.5 Additional UPnP-Related Extensions**

3396 In addition to supporting SSDP-based discovery, NEMO nodes may optionally support additional  
3397 extensions that make interacting with a UPnP environment more seamless. Nodes supporting this  
3398 binding will comply with the protocols for supporting addressing, discovery, and description, as  
3399 set forth in the UPnP Device Architecture Version 1.0 [\[UPnP\]](#) specifications.

3400 These proposed extensions allow a NEMO nodes to be exposed as a UPnP device. The NEMO  
3401 node may be represented as a virtual device or may correspond to an actual physical device. In  
3402 particular, these extensions:

- 3403       1. Allow NEMO nodes to expose services that fit the traditional UPnP usage model as  
3404       UPnP services. A NEMO node may choose to expose and describe services that are  
3405       completely compliant with traditional UPnP usage models, or not.
- 3406       2. Allow NEMO nodes to expose information about the node and about services that are  
3407       beyond the current scope of UPnP usage models. For example, certain services may  
3408       require authentication and authorization capabilities that aren't directly supported by  
3409       UPnP protocols.
- 3410       3. Allow non-NEMO node UPnP devices to discover NEMO nodes and their services. A  
3411       standard UPnP device should have the capability to discover NEMO nodes and their  
3412       extended properties. A UPnP device may be unaware of or choose not to process any  
3413       properties and services that don't fit the standard usage model, but the capability exists  
3414       for discovery.
- 3415       4. Allow NEMO nodes to discover, inspect, and invoke services on non-NEMO node-ed  
3416       UPnP devices. In this case, the NEMO node will act in a capacity that supports the UPnP  
3417       control point.

3418 In this binding we deal with the following cases:

- 3419       1. One-to-one correspondence between NEMO node and actual device, i.e., a device fully  
3420       encapsulates a NEMO node. In this case, the `upnp:rootdevice` type and associated  
3421       device UUID will be bound to the device itself.
- 3422       2. NEMO node without corresponding device. In this case, the NEMO node will represent  
3423       itself as a UPnP root device with appropriate device type, UUID and other necessary  
3424       information to describe it.

3425 3. NEMO nodes may act as UPnP control points for the purposes of actually performing  
3426 searches or receiving events.

3427 In terms of protocol support more specifically:

3428 1. NEMO nodes will support UPnP discovery advertisement, and when a node is added to a  
3429 network, the UPnP discovery protocol will allow that node to advertise its services to  
3430 UPnP control points.

3431 2. NEMO nodes will support discovery search requests for M-Search and respond  
3432 accordingly.

3433 3. NEMO nodes may expose a UPnP device description that contains a set of extensions for  
3434 supporting properties and services that are beyond standard UPnP usage models.

### 3435 **6.2.5.1 Extensions to Device Description**

3436 As previously mentioned, a NEMO node may be inspected after being discovered using SSDP.  
3437 The information related to inspection of the node is in the form of an XML encoded device  
3438 description, obtained via HTTP, whose location is communicated back in a discovery response.  
3439 In addition to the standard UPnP information contained in the description, a NEMO node may  
3440 include an extension that defines additional information.

3441 The extension type X\_NEMONode-Extension in the `http://nemo.intertrust.com/discovery/upnp`  
3442 namespace defines properties specifically related to NEMO nodes and services that fall outside  
3443 the scope of the traditional UPnP usage model.

3444 In particular, the extension may include:

3445 1. NEMO node info, which can include node ID and security credentials related to the  
3446 node, such as SAML roles defined for the role and public keys associated with usage of  
3447 services.

3448 2. List of services defined for the node, including an endpoint reference for contacting the  
3449 service. We use WS-Addressing Endpoint references to describe the service endpoints.

3450 3. A description of how to further inspect the services that are specified in the extension.  
3451 Here we leverage the existing mechanism we defined for inspection with WS-Discovery,  
3452 providing an inspection reference (`nemoc:InspectionReference`), which could  
3453 contain a pointer to where the WSDL could be obtained or a pointer to a service that  
3454 could be used for further inspection, such as inspection services based on WS-  
3455 MetadataExchange.

3456 Section 6.4.3.1 describes the schemas for the device description extension and §6.5.7 provides an  
3457 example descriptor.

### 3458 **6.2.5.2 Extensions to Service Description**

3459 A NEMO node is required to expose one or more services to be UPnP compliant. The  
3460 descriptions of these services will be exposed to other UPnP entities in the form of SCP  
3461 descriptors.

3462 Some form of the SCP descriptor may vary depending on the intended usage. The following are  
3463 three potential forms:

- 3464 1. Standard SCPD with both UPnP service actions and events. This is the form of descriptor  
3465 that today is commonly used with UPnP entities.
- 3466 2. SCPD with no actions or events and with a NEMO service extension. This is the form of  
3467 descriptor where it is not possible to describe the NEMO service input and output  
3468 messages based on the standard SCP description language. The form supports discovery  
3469 of the service over UPnP, but further understanding of the service requirements will  
3470 either have to be made through local means or via processing of the NEMO service  
3471 extension.
- 3472 3. SCPD with actions and events and with a NEMO service extension. This is the form of  
3473 the service description where we support the traditional UPnP usage model but include  
3474 additional service information in the extension that could be used by clients that are  
3475 NEMO-aware, but it is not mandatory across all clients.

3476 The extension type X\_NEMOService-Extension in the <http://nemo.intertrust.com/discovery/upnp>  
3477 namespace defines properties specifically related to a NEMO node's service that fall outside the  
3478 scope of the traditional UPnP usage model.

3479 In particular, the extension may include:

3480 Reference to a WSDL description or the actual WSDL itself for a service if it cannot be  
3481 adequately described in SCP.

3482 A description of how to further inspect the service that is being specified. Here we leverage  
3483 the existing mechanism we defined for inspection with WS-Discovery, providing an  
3484 inspection reference (nemoc:InspectionReference).

3485 Section 6.4.3.2 describes the schemas for the service description extension, and §6.5.8 provides  
3486 an example descriptor.

### 3487 **6.2.5.3 UPnP Service Security**

3488 NEMO services are secured based on extensible security protocols in the web services  
3489 community.

3490 Even though most standard UPnP environments either lack general mechanisms for service level  
3491 security or secure the contents of a service message at the business level payload, there do exist  
3492 proposals for standard UPnP based service security, including the specifications based on Device  
3493 Security and Security Console V 1.0 (<http://www.upnp.org/standardizeddcps/security.asp>).

## 3494 **6.3 Inspection Binding**

### 3495 **6.3.1 Description**

3496 Given a reference to a NEMO node, inspection provides the ability to query it about certain well-  
3497 defined attributes (metadata), such as the descriptions of the policy related to the services it  
3498 publicly offers or WSDL or other more general NEMO node information.

3499 In this current binding we are currently addressing inspection of metadata surrounding target  
3500 service endpoints exported by NEMO nodes. Inspection is supported as a well-defined service a  
3501 node exports.

## 3502 6.3.2 Requirements (Normative)

3503 NEMO nodes supporting this binding and offering services shall comply with the Web Services  
3504 Metadata Exchange [\[WS-MetadataExchange\]](#) specification. As stated in the specification, the  
3505 following message protocols will be supported:

- 3506 • MetadataGet: a node initiates a Metadata GET request, specifying a target service  
3507 endpoint and potentially a dialect and/or identifier which designate the specific type of  
3508 metadata or specific metadata object to retrieve. In response, the requested metadata for  
3509 the target service endpoint is returned in terms of its actual content and/or a reference to  
3510 where to obtain the content via a GET operation. If no dialect or identifier is provided,  
3511 all metadata for the target service endpoint is returned.
- 3512 • Get: a node initiates a GET Request specifying a reference to a specific instance of  
3513 metadata. In response, the actual metadata is returned.

3514 In addition to the standard types of metadata dialects for WSDL and Policy, we define a new  
3515 dialect for retrieving NEMO node information related to a given service address:

3516 `http://nemo.intertrust.com/2004/inspection/mex/nemonodeinfo`

3517 At a minimum, NEMO nodes **may** understand and support retrieving metadata formats based on  
3518 the following standard dialects:

- 3519 • XML Schema Version 1.0
- 3520 • WSDL 1.1
- 3521 • WS-Policy expression

3522 Additional dialects may also be defined, such as:

- 3523 • NEMO Node Information

## 3524 6.4 Schema and Abstract Web Service Definitions

### 3525 6.4.1 NEMO Node Description

```
3526 <xsd:schema targetNamespace="http://www.intertrust.com/core"  
3527 nemoc="http://www.intertrust.com/core" xmlns:wss="http://docs.oasis-  
3528 open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"  
3529 elementFormDefault="qualified" attributeFormDefault="qualified"  
3530 version="0.5">  
3531 <xsd:complexType name="NodeInfo">  
3532 <xsd:complexContent>  
3533 <xsd:extension base="nemoc:Base">  
3534 <xsd:sequence>  
3535 <xsd:element name="NodeId" type="xsd:uri" minOccurs="0"/>  
3536 <xsd:element ref="wss:SecurityTokenReference" minOccurs="0"  
3537 maxOccurs="unbounded"/>  
3538 </xsd:sequence>  
3539 </xsd:extension>  
3540 </xsd:complexType>
```

```
3542 </xsd:complexType>
3543 <xsd:element name="NodeInfo" type="nemoc:NodeInfo"/>
3544 </xsd:schema>
```

## 3545 6.4.2 Discovery

### 3546 6.4.2.1 WS-Discovery Scope-Related Definitions

```
3547
3548 <xsd:schema targetNamespace="http://nemo.intertrust.com/discovery/wsd"
3549 nemodisc="http://nemo.intertrust.com/discovery/wsd"
3550 nemoc="http://www.intertrust.com/core"
3551 xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
3552 elementFormDefault="qualified" attributeFormDefault="qualified"
3553 version="0.5">
3554
3555 <xsd:complexType name="RoleAttributeDesignator">
3556 <xsd:complexContent>
3557 <xsd:extension base="nemoc:Base">
3558 <xsd:sequence>
3559 <xsd:element name="issuer" type="xsd:string" minOccurs="0">
3560 <xsd:element name="name" type="xsd:string">
3561 <xsd:element name="value" type="xsd:string">
3562 </xsd:sequence>
3563 </xsd:extension>
3564 </xsd:complexContent>
3565 </xsd:complexType>
3566 <xsd:element name="RoleAttributeDesignator"
3567 type="nemodisc:RoleAttributeDesignator"/>
3568
3569 <xsd:complexType name="RoleScopeCriteria">
3570 <xsd:complexContent>
3571 <xsd:extension base="nemoc:Base">
3572 <xsd:sequence>
3573 <xsd:element name="targettype" type="xsd:string"> <!-- NODE,
3574 SERVICE -->
3575 <xsd:element ref="nemodisc:RoleAttributeDesignator"
3576 minOccurs="1" maxOccurs="unbounded">
3577 </xsd:sequence>
3578 </xsd:extension>
3579 </xsd:complexContent>
3580 </xsd:complexType>
3581 <xsd:element name="RoleScopeCriteria"
3582 type="nemodisc:RoleScopeCriteria"/>
3583
3584
3585 <xsd:complexType name="PolicyTokenScopeCriteria">
3586 <xsd:complexContent>
3587 <xsd:extension base="nemoc:Base">
3588 <xsd:sequence>
3589 <xsd:element ref="wsse:SecurityTokenReference">
3590 </xsd:sequence>
```

```

3591     </xsd:extension>
3592 </xsd:complexContent>
3593 </xsd:complexType>
3594 <xsd:element name="PolicyTokenScopeCriteria"
3595 type="nemodisc:PolicyTokenScopeCriteria"/>
3596
3597 </xsd:schema>

```

3598

## 3599 **6.4.2.2 WS-Discovery Discovery Proxy Schema Definitions** 3600 **and WSDL**

3601

```

3602 <wsdl:definitions
3603     xmlns="http://schemas.xmlsoap.org/wsdl/"
3604     targetNamespace='http://nemo.intertrust.com/2004/discovery'
3605     xmlns:tns="http://nemo.intertrust.com/2004/discovery"
3606     xmlns:ds='http://schemas.xmlsoap.org/ws/2004/02/discovery'
3607     xmlns:wSDL='http://schemas.xmlsoap.org/wsdl/'
3608     xmlns:wSDLsoap='http://schemas.xmlsoap.org/wsdl/soap/'
3609     xmlns:xs='http://www.w3.org/2001/XMLSchema'
3610     name="DiscoveryProxyService">
3611
3612     <wsdl:types>
3613         <xs:schema xmlns="http://nemo.intertrust.com/2004/discovery"
3614         targetNamespace="http://nemo.intertrust.com/2004/discovery">
3615             <xs:import
3616             namespace='http://schemas.xmlsoap.org/ws/2004/02/discovery'
3617             location='discovery.xsd' />
3618             <xs:element name="ProxyProbe">
3619                 <xs:complexType>
3620                     <xs:sequence>
3621                         <xs:element ref="ds:Probe"/>
3622                     </xs:sequence>
3623                 </xs:complexType>
3624             </xs:element>
3625             <xs:element name="ProxyProbeMatch">
3626                 <xs:complexType>
3627                     <xs:sequence>
3628                         <xs:element ref="ds:ProbeMatches minOccurs="0"/>
3629                     </xs:sequence>
3630                 </xs:complexType>
3631             </xs:element>
3632             <xs:element name="ProxyResolveMatch">
3633                 <xs:complexType>
3634                     <xs:sequence>
3635                         <xs:element ref="ds:ResolveMatch" minOccurs="0"/>
3636                     </xs:sequence>
3637                 </xs:complexType>
3638             </xs:element>
3639         </xs:schema>
3640     </wsdl:types>

```

```

3641
3642 <wsdl:message name="ProxyProbeMsg">
3643   <wsdl:part name='body' element="tns:ProxyProbe" />
3644 </wsdl:message>
3645
3646 <wsdl:message name="ProxyProbeMatchMsg">
3647   <wsdl:part name='body' element="tns:ProxyProbeMatch" />
3648 </wsdl:message>
3649
3650 <wsdl:message name="ProxyResolveMsg">
3651 </wsdl:message>
3652
3653 <wsdl:message name="ProxyResolveMatchMsg">
3654   <wsdl:part name='body' element="tns:ProxyResolveMatch" />
3655 </wsdl:message>
3656
3657
3658 <wsdl:portType name="DiscoveryProxy">
3659   <wsdl:operation name='ProxyProbeOp'>
3660     <wsdl:input message="tns:ProxyProbeMsg" />
3661     <wsdl:output message="tns:ProxyProbeMatchMsg" />
3662   </wsdl:operation>
3663   <wsdl:operation name='ProxyResolveOp'>
3664     <wsdl:input message="tns:ProxyResolveMsg" />
3665     <wsdl:output message="tns:ProxyResolveMatchMsg" />
3666   </wsdl:operation>
3667 </wsdl:portType>
3668
3669 </wsdl:definitions>

```

### 3670 **6.4.3 Inspection**

#### 3671 **6.4.3.1 NEMO UPnP Device Description Extension**

```

3672
3673 <xsd:schema targetNamespace=http://nemo.intertrust.com/discovery/upnp
3674 nemoupnp="http://nemo.intertrust.com/discovery/upnp"
3675 nemoc="http://www.intertrust.com/core"
3676 xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
3677 elementFormDefault="qualified" attributeFormDefault="qualified"
3678 version="0.5">
3679
3680 <xsd:complexType name="ServiceList">
3681   <xsd:complexContent>
3682     <xsd:extension base="nemoc:Base">
3683       <xsd:sequence>
3684         <xsd:element ref="nemoc:InspectionReference" minOccurs="0">
3685         <xsd:element ref="wsa:EndpointReference"
3686 minOccurs="0" maxOccurs="unbounded">
3687       </xsd:sequence>
3688     </xsd:extension>
3689   </xsd:complexContent>

```

```

3690 </xsd:complexType>
3691 <xsd:element name="ServiceList" type="nemoupnp:ServiceList"/>
3692
3693 <xsd:complexType name="X_NEMONode-Extension">
3694 <xsd:complexContent>
3695 <xsd:extension base="nemoc:Base">
3696 <xsd:sequence>
3697 <xsd:element ref="nemoc:NodeInfo">
3698 <xsd:element ref="nemoupnp:ServiceList" minOccurs="0">
3699 </xsd:sequence>
3700 </xsd:extension>
3701 </xsd:complexContent>
3702 </xsd:complexType>
3703 <xsd:element name="X_NEMONode-Extension" type="nemoupnp:X_NEMONode-
3704 Extension"/>
3705 </xsd:schema>

```

### 3706 **6.4.3.2 NEMO UPnP Service Description Extension**

```

3707
3708 <xsd:schema targetNamespace=http://nemo.intertrust.com/discovery/upnp
3709 nemoupnp="http://nemo.intertrust.com/discovery/upnp"
3710 nemoc="http://www.intertrust.com/core"
3711 xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
3712 xmlns:wSDL="http://schemas.xmlsoap.org/wSDL"
3713 elementFormDefault="qualified" attributeFormDefault="qualified"
3714 version="0.5">
3715
3716 <xsd:complexType name="X_NEMOService-Extension">
3717 <xsd:complexContent>
3718 <xsd:extension base="nemoc:Base">
3719 <xsd:sequence>
3720 <xsd:element ref="wSDL:definitions" minOccurs="0">
3721 <xsd:element ref="nemoc:InspectionReference"
3722 minOccurs="0">
3723 </xsd:sequence>
3724 </xsd:extension>
3725 </xsd:complexContent>
3726 </xsd:complexType>
3727 <xsd:element name="X_NEMOService-Extension"
3728 type="nemoupnp:X_NEMOService-Extension"/>
3729 </xsd:schema>

```

## 3730 **6.5 Sample Messages**

### 3731 **6.5.1 WS-Discovery Probe for a Service that Supports** 3732 **DiscoveryProxy.**

```

3733
3734 <soap:Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/"
3735 xmlns:nemo="http://nemo.intertrust.com/2004/core">

```

```

3736 xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
3737 xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
3738 xmlns:wsd="http://schemas.xmlsoap.org/ws/2004/10/discovery"
3739 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
3740 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
3741
3742 <soap:Header>
3743   <wsa:Action>
3744     http://schemas.xmlsoap.org/ws/2004/10/discovery/Probe
3745   </wsa:Action>
3746   <wsa:MessageID>uuid:1097696212363</wsa:MessageID>
3747   <wsa:To>urn:schemas-xmlsoap-org:ws:2004:10:discovery</wsa:To>
3748   <wsa:ReplyTo>
3749     <wsa:Address>
3750       http://localhost:9080/SimpleDiscovery/services/Discovery
3751     </wsa:Address>
3752   </wsa:ReplyTo>
3753 </soap:Header>
3754
3755 <soap:Body>
3756 <wsd:Probe>
3757 <wsd:Types xmlns:dtyens0="http://nemo.intertrust.com/2004/discovery">
3758 dtyens0:DiscoveryProxy
3759 </wsd:Types>
3760 </wsd:Probe>
3761 </soap:Body>
3762 </soap:Envelope>

```

3763

## 3764 **6.5.2 WS-Discovery ProbeMatch for a Service that** 3765 **Supports DiscoveryProxy.**

```

3766
3767 <?xml version="1.0" encoding="UTF-8"?>
3768 <soap:Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/"
3769 xmlns:nemo="http://nemo.intertrust.com/2004/core"
3770 xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
3771 xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
3772 xmlns:wsd="http://schemas.xmlsoap.org/ws/2004/10/discovery"
3773 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
3774 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
3775
3776 <soap:Header>
3777   <wsa:Action>
3778     http://schemas.xmlsoap.org/ws/2004/10/discovery/ProbeMatch
3779   </wsa:Action>
3780   <wsa:MessageID>uuid:1097696212463</wsa:MessageID>
3781   <wsa:RelatesTo>uuid:1097696212363</wsa:RelatesTo>
3782   <wsa:To>http://localhost:9080/SimpleDiscovery/services/Discovery
3783   </wsa:To>
3784 </soap:Header>

```

```

3785
3786 <soap:Body>
3787 <wsd:ProbeMatches>
3788   <wsd:ProbeMatch>
3789     <wsa:EndpointReference>
3790       <wsa:Address>
3791         http://localhost:9084/SimpleDiscovery/services/DiscoveryProxy
3792       </wsa:Address>
3793       <wsa:ReferenceParameters>
3794         <nemo:InspectionReference>
3795           xmlns:nemo=http://nemo.intertrust.com/2004/core>
3796           <wsa:EndpointReference>
3797             <wsa:Address>
3798               http://localhost:9084/SimpleDiscovery/services/
3799 MetadataExchange
3800             </wsa:Address>
3801           </wsa:EndpointReference>
3802         </nemo:InspectionReference>
3803       </wsa:ReferenceParameters>
3804       <wsa:ReferenceProperties>
3805         <nemo:NodeInfo>
3806           xmlns:nemo="http://nemo.intertrust.com/2004/core">
3807             <nemo:NodeId>
3808               xmlns:nemo="http://nemo.intertrust.com/2004/core">
3809 urn:node002
3810             </nemo:NodeId>
3811           </nemo:NodeInfo>
3812         </wsa:ReferenceProperties>
3813       <wsa:PortType>
3814         xmlns:svcns0="http://nemo.intertrust.com/2004/discovery">
3815         svcns0:DiscoveryProxy
3816       </wsa:PortType>
3817       <wsa:ServiceName>
3818         xmlns:svcns0="http://nemo.intertrust.com/2004/discovery">
3819         svcns0:DiscoveryProxyService
3820       </wsa:ServiceName>
3821     </wsa:EndpointReference>
3822   <wsd:Types>
3823     xmlns:dtypens0="http://nemo.intertrust.com/2004/discovery">
3824     dtypens0:DiscoveryProxy
3825   </wsd:Types>
3826   <wsd:Scope>
3827     MatchBy="http://nemo.intertrust.com/discovery/scope/matchbynodeinfo">
3828       <nemo:NodeInfo>
3829         xmlns:nemo="http://nemo.intertrust.com/2004/core">
3830           <nemo:NodeId>
3831             xmlns:nemo="http://nemo.intertrust.com/2004/core">
3832             urn:node002
3833           </nemo:NodeId>
3834         </nemo:NodeInfo>
3835       </wsd:Scope>
3836   </wsd:XAddr>

```

```

3837 http://localhost:9084/SimpleDiscovery/services/DiscoveryProxy
3838     </wsd:XAddrs>
3839     <wsd:MetadataVersion>1</wsd:MetadataVersion>
3840     </wsd:ProbeMatch>
3841 </wsd:ProbeMatches>
3842 </soap:Body>
3843 </soap:Envelope>

```

### 3844 **6.5.3 WS-Discovery Hello Announcement to Announce** 3845 **Availability of Service**

```

3846 <soap:Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/"
3847 xmlns:nemo="http://nemo.intertrust.com/2004/core"
3848 xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
3849 xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
3850 xmlns:wsd="http://schemas.xmlsoap.org/ws/2004/10/discovery"
3851 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
3852 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
3853
3854 <soap:Header>
3855     <wsa:Action>
3856 http://schemas.xmlsoap.org/ws/2004/10/discovery/Hello
3857     </wsa:Action>
3858     <wsa:MessageID>uuid:1097695419415</wsa:MessageID>
3859     <wsa:To>urn:schemas-xmlsoap-org:ws:2004:10:discovery</wsa:To>
3860     <wsd:AppSequence InstanceId="00032111111" MessageNumber="1"/>
3861 </soap:Header>
3862
3863 <soap:Body>
3864 <wsd:Hello>
3865     <wsa:EndpointReference>
3866         <wsa:Address>
3867 http://localhost:9080/SimpleDiscovery/services/TestE
3868         </wsa:Address>
3869         <wsa:ReferenceParameters>
3870             <nemo:InspectionReference
3871 xmlns:nemo="http://nemo.intertrust.com/2004/core">
3872                 <wsa:EndpointReference>
3873                     <wsa:Address>
3874 http://localhost:9080/SimpleDiscovery/services/MetadataExchange
3875                     </wsa:Address>
3876                 </wsa:EndpointReference>
3877             </nemo:InspectionReference>
3878         </wsa:ReferenceParameters>
3879         <wsa:ReferenceProperties>
3880             <nemo:NodeInfo
3881 xmlns:nemo="http://nemo.intertrust.com/2004/core">
3882                 <nemo:NodeId
3883 xmlns:nemo="http://nemo.intertrust.com/2004/core">
3884 urn:node000
3885                 </nemo:NodeId>
3886

```

```

3887         </nemo:NodeInfo>
3888     </wsa:ReferenceProperties>
3889     <wsa:PortType xmlns:svcns0="http://www.intertrust.com/services">
3890 svcns0:TestE</wsa:PortType>
3891     <wsa:ServiceName
3892 xmlns:svcns0="http://www.intertrust.com/services">
3893 svcns0:TestEService
3894     </wsa:ServiceName>
3895 </wsa:EndpointReference>
3896 <wsd:Types xmlns:dtypens0="http://www.intertrust.com/services">
3897 dtypens0:TestE
3898 </wsd:Types>
3899 <wsd:Scope
3900 MatchBy="http://nemo.intertrust.com/discovery/scope/matchbynodeinfo">
3901     <nemo:NodeInfo xmlns:nemo="http://nemo.intertrust.com/2004/core">
3902     <nemo:NodeId xmlns:nemo="http://nemo.intertrust.com/2004/core">
3903 urn:node000
3904     </nemo:NodeId>
3905     </nemo:NodeInfo>
3906 </wsd:Scope>
3907 <wsd:XAddrs>
3908 http://localhost:9080/SimpleDiscovery/services/TestE
3909 </wsd:XAddrs>
3910 <wsd:MetadataVersion>1</wsd:MetadataVersion>
3911 </wsd:Hello>
3912 </soap:Body>
3913 </soap:Envelope>

```

## 3914 **6.5.4 WS-Discovery Bye to Announce Service Is No Longer** 3915 **Available**

```

3916 <soap:Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/"
3917 xmlns:nemo="http://nemo.intertrust.com/2004/core"
3918 xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
3919 xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
3920 xmlns:wsd="http://schemas.xmlsoap.org/ws/2004/10/discovery"
3921 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
3922 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
3923
3924 <soap:Header>
3925     <wsa:Action>http://schemas.xmlsoap.org/ws/2004/10/discovery/Bye</wsa:
3926 Action>
3927     <wsa:MessageID>uuid:1097695958833</wsa:MessageID>
3928 <wsa:To>urn:schemas-xmlsoap-org:ws:2004:10:discovery</wsa:To>
3929     <wsd:AppSequence InstanceId="00032111111" MessageNumber="1"/>
3930 </soap:Header>
3931
3932 <soap:Body>
3933 <wsd:Bye>
3934     <wsa:EndpointReference>
3935     <wsa:Address>

```

```

3937 http://localhost:9084/SimpleDiscovery/services/TestA</wsa:Address>
3938     <wsa:ReferenceParameters>
3939         <nemo:InspectionReference
3940 xmlns:nemo="http://nemo.intertrust.com/2004/core">
3941             <wsa:EndpointReference>
3942                 <wsa:Address>
3943 http://localhost:9084/SimpleDiscovery/services/MetadataExchange
3944                 </wsa:Address>
3945             </wsa:EndpointReference>
3946         </nemo:InspectionReference>
3947     </wsa:ReferenceParameters>
3948 <wsa:ReferenceProperties>
3949     <nemo:NodeInfo
3950 xmlns:nemo="http://nemo.intertrust.com/2004/core">
3951         <nemo:NodeId
3952 xmlns:nemo="http://nemo.intertrust.com/2004/core">
3953 urn:node002
3954         </nemo:NodeId>
3955     </nemo:NodeInfo>
3956 </wsa:ReferenceProperties>
3957 </wsa:EndpointReference>
3958 </wsd:Bye>
3959 </soap:Body>
3960 </soap:Envelope>

```

## 3961 **6.5.5 WS-MetadataExchange Request to Obtain a Service**

### 3962 **WSDL**

```

3963
3964 <soap:Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/"
3965 xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
3966 xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
3967 xmlns:wsx="http://schemas.xmlsoap.org/ws/2004/09/mex"
3968 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
3969 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
3970
3971 <soap:Header>
3972     <wsa:Action>
3973 http://schemas.xmlsoap.org/ws/2004/09/mex/GetMetadata/Request
3974     </wsa:Action>
3975     <wsa:MessageID>uuid:1097698812758</wsa:MessageID>
3976     <wsa:To>http://localhost:9082/SimpleDiscovery/services/TestB</wsa:To>
3977 </soap:Header>
3978
3979 <soap:Body>
3980 <wsx:GetMetadata>
3981     <wsx:Dialect>http://schemas.xmlsoap.org/wSDL/</wsx:Dialect>
3982 </wsx:GetMetadata>
3983 </soap:Body>
3984 </soap:Envelope>

```

3985 **6.5.6 WS-MetadataExchange Response for Obtained**  
3986 **Service WSDL**

```
3987
3988 <?xml version="1.0" encoding="UTF-8"?>
3989 <soap:Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/"
3990 xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
3991 xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
3992 xmlns:wsx="http://schemas.xmlsoap.org/ws/2004/09/mex"
3993 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
3994 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
3995
3996 <soap:Header>
3997   <wsa:Action>
3998 http://schemas.xmlsoap.org/ws/2004/09/mex/GetMetadata/Response
3999   </wsa:Action>
4000   <wsa:MessageID>uuid:1097698813489</wsa:MessageID>
4001   <wsa:RelatesTo>uuid:1097698812758</wsa:RelatesTo>
4002 </soap:Header>
4003
4004 <soap:Body>
4005 <wsx:Metadata>
4006 <wsx:MetadataSectionDialect="http://schemas.xmlsoap.org/wsd1/">
4007 <wsdl:definitions name="TestBService"
4008 targetNamespace="http://www.intertrust.com/services"
4009 xmlns="http://schemas.xmlsoap.org/wsd1/"
4010 xmlns:apachesoap="http://xml.apache.org/xml-soap"
4011 xmlns:impl="http://www.intertrust.com/services"
4012 xmlns:intf="http://www.intertrust.com/services"
4013 xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
4014 xmlns:tnstype="http://www.intertrust.com/services"
4015 xmlns:wsd1="http://schemas.xmlsoap.org/wsd1/"
4016 xmlns:wsd1soap="http://schemas.xmlsoap.org/wsd1/soap/"
4017 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
4018
4019
4020   <!-- =====
4021   You can define additional types using the <wsdl:types>
4022   element, as shown below.
4023   ===== -->
4024   <wsdl:types>
4025   </wsdl:types>
4026
4027
4028   <!-- message declarations -->
4029
4030   <wsdl:message name="anydata">
4031     <wsdl:part element="xsd:string" name="body"/>
4032   </wsdl:message>
4033
4034   <!-- port type declarations -->
4035   <wsdl:portType name="TestB">
```

```

4036     <wsdl:operation name="invoke">
4037         <wsdl:input message="impl:anydata"/>
4038         <wsdl:output message="impl:anydata"/>
4039     </wsdl:operation>
4040 </wsdl:portType>
4041
4042 <!-- binding declarations -->
4043 <wsdl:binding name="TestBSoapBinding" type="impl:TestB">
4044     <wsdlsoap:binding style="document"
4045 transport="http://schemas.xmlsoap.org/soap/http"/>
4046     <wsdl:operation name="invoke">
4047         <wsdlsoap:operation
4048 soapAction="http://www.intertrust.com/services/TestB/invoke"/>
4049         <wsdl:input name="invokeRequest">
4050             <wsdlsoap:body use="literal"/>
4051         </wsdl:input>
4052         <wsdl:output name="invokeResponse">
4053             <wsdlsoap:body use="literal"/>
4054         </wsdl:output>
4055     </wsdl:operation>
4056 </wsdl:binding>
4057
4058 <!-- service declarations -->
4059 <wsdl:service name="TestBService">
4060     <wsdl:port binding="impl:TestBSoapBinding" name="TestB">
4061         <wsdlsoap:address
4062 location="http://localhost:9082/SimpleDiscovery/services/TestB"/>
4063     </wsdl:port>
4064 </wsdl:service>
4065
4066 </wsdl:definitions>
4067 </wsx:MetadataSection>
4068 </wsx:Metadata>
4069 </soap:Body>
4070 </soap:Envelope>
4071
4072

```

## 6.5.7 Example UPnP Device Description with NEMO Extension

```

4073
4074
4075
4076 <root xmlns="urn:schemas-upnp-org:device-1-0">
4077     <specVersion>
4078         <major>1</major>
4079         <minor>0</minor>
4080     </specVersion>
4081     <device>
4082         <deviceType>urn:schemas-upnp-org:device:simple:1</deviceType>
4083         <friendlyName>Generic Simple Device</friendlyName>
4084         <manufacturer>Intertrust</manufacturer>
4085         <manufacturerURL>http://www.intertrust.com</manufacturerURL>

```

```

4086     <modelDescription>Generic Simple Device</modelDescription>
4087     <modelName>Simple</modelName>
4088     <modelNumber>1.0</modelNumber>
4089     <modelURL>http://www.intertrust.com</modelURL>
4090     <serialNumber>1234567890</serialNumber>
4091     <UDN>uuid:simpledevice</UDN>
4092     <UPC>123456789012</UPC>
4093
4094 <serviceList>
4095 <service>
4096     <serviceType>urn:intertrust:service:NEMO-
4097 LicenseCreator:1</serviceType>
4098     <serviceId>urn:intertrust:serviceId:LicenseCreator</serviceId>
4099     <SCPDURL>/LicenseCreator.xml</SCPDURL>
4100     <ControlURL>/LicenseCreator.xml</ControlURL>
4101     <eventSubURL></eventSubURL>
4102 </service>
4103 </serviceList>
4104
4105 <X_NEMONode-Extension xmlns="http://nemo.intertrust.com/discovery/upnp">
4106
4107 <nemoc:NodeInfo>
4108     <nemoc:NodeId>urn:nemo:LicenseCreator</nemoc:NodeId>
4109     <!-- public encryption Key associated with all node services -->
4110     <wsse:SecurityTokenReference
4111 nemosec:Usage="http://nemo.intertrust.com/2004/security/secure-
4112 protocol/basic/1.0#request-encryptionKey">
4113         <wsse:Embedded>
4114             <wsse:BinarySecurityToken
4115                 ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
4116 wss-x509-token-profile-1.0#X509PKIPathv1"
4117                 EncodingType="wsse:Base64Binary">
4118 MIICFzCCAhMwggF8oAMCAQICBEGRH+cwDQYJKoZIhvcNAQEFBQAwHzEdMBsGCisG
4119 AQQBuyABAQETDXVybjpuZW1vOlJvb3QwHhcNMDQxMTA5MTk1MjA3WhcNMDUxMTA5
4120 MTk1MjA3WjApMScWJQYKKWYBBAG7IAEBARMXdxJuOm5lbW86TG1jZW5zZUNyZWZ0
4121 b3IwZz8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAKS9veyG4L5b4pY9+uRdvOaS
4122 HsURSrYfGsgPo8yLHb2hBVj2R0AlB4N6+ZZbuqJTgh7Y1mIimjRM/zgMkIpirm9m
4123 8FpiPKRcCm5kA8f5tw3LXbx1LBCp2z5v36q/mSDFLtbEph0Jd5QRlGRgsME+46DR
4124 LYygVEzSak8s1Muw0tHNAgMBAAGjUjBQMB8GA1UdIwQYMBaAFEFHN1ot9Wtw8fLy
4125 GuCZobAF006CMB0GA1UdDgQWBBSbaOB30xjx80f1qzxAHaS948XjbzAOBgNVHQ8B
4126 Af8EBAMCBSAwDQYJKoZIhvcNAQEFBQADgYEAj0UtfTx7bmCXISvwIcswVfK8R46
4127 Foh6sbn2khEplqQkWA6BggRL0tjQQ7DICKcEHSOF+eZrNC+8zUYRGR81Lnysaqj8
4128 WXMNjFUXIZMqCiThyrDP6ZbCsUcCABoo/emSyf5wGzLjo6BnGeP+cfMTJ/7iSxY6
4129 TZG2IkVWrfULbVM=
4130             </wsse:BinarySecurityToken>
4131         </wsse:Embedded>
4132     </wsse:SecurityTokenReference>
4133
4134     <ServiceList>
4135
4136     <nemoc:InspectionReference
4137 xmlns:nemoc="http://nemo.intertrust.com/2004/core">

```

```

4138     <wsa:EndpointReference>
4139
4140     <wsa:Address>http://localhost:9080/services/MetadataExchange</wsa:Add
4141     ress>
4142     </wsa:EndpointReference>
4143     </nemoc:InspectionReference>
4144
4145     <wsa:EndpointReference>
4146     <wsa:Address>
4147     http://localhost:9080/services/LicenseCreator</wsa:Address>
4148     <wsa:PortType
4149     xmlns:svcns0="http://www.intertrust.com/services">
4150     svcns0:LicenseCreator</wsa:PortType>
4151     <wsa:ServiceName
4152     xmlns:svcns0="http://www.intertrust.com/services">
4153     svcns0:LicenseCreator</wsa:ServiceName>
4154     </wsa:EndpointReference>
4155     </ServiceList>
4156 </nemoc:NodeInfo>
4157 </X_NEMONode-Extension>
4158 </device>
4159 </root>

```

## 4160 6.5.8 Example UPnP Service Description with NEMO 4161 Extension

```

4162 <scpd xmlns:"urn:intertrust:service-1-0">
4163   <specVersion>
4164     <major>1</major>
4165     <minor>0</minor>
4166   </specVersion>
4167   <serviceStateTable>
4168     <stateVariable sendEvents="no">
4169       <name>X_NEMO_Node_ID</name>
4170       <datatype>uuid</datatype>
4171     </stateVariable>
4172   </serviceStateTable>
4173   <X_NEMOService-Extension
4174     xmlns="http://www.intertrust.com/discovery/upnp"
4175     xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
4176     <nemoc:InspectionReference
4177       xmlns:nemoc="http://nemo.intertrust.com/2004/core">
4178       <wsa:EndpointReference>
4179         <wsa:Address>
4180           http://www.intertrust.com/services/LicenseCreator.wsdl
4181         </wsa:Address>
4182       </wsa:EndpointReference>
4183     </nemoc:InspectionReference>
4184   </X_NEMOService-Extension>
4185 </scpd>

```

## 4187 **6.5.9 Example SSDP Discovery Request Message**

```
4188  
4189 M-SEARCH * HTTP/1.1  
4190 S: uuid:ijklmnop-7dec-11d0-a765-00a0c91e6bf6  
4191 Host: 239.255.255.250:reservedSSDPport  
4192 Man: "ssdp:discover"  
4193 ST: http://nemo.intertrust.com/services#OctopusLicenseService  
4194 MX: 3
```

## 4195 **6.5.10 Example SSDP Discovery Response Message**

```
4196  
4197 HTTP/1.1 200 OK  
4198 S: uuid:ijklmnop-7dec-11d0-a765-00a0c91e6bf6  
4199 Ext:  
4200 Cache-Control: no-cache="Ext", max-age = 5000  
4201 ST: http://nemo.intertrust.com/services#OctopusLicenseService  
4202 USN: http://localhost:9082/MarlinSettop/services/OctopusLicense  
4203 Location: http://localhost:8080/service/inspection  
4204 Inspection-Policy-Id: urn:nemopolicyid:00001
```

4205

## 7 References (Normative)

4206

The following table lists references cited in this document.

4207

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

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

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

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

4208