



1

OSCI-Transport, Version 2.0

2

3

– Web Services Profiling and Extensions Specification –

4

OSCI Steering Office

5

Status: Final

6

Edition 1, published 4th of June, 2009

7 This is the approved final version of the OSCI 2.0 Web Services Profiling and Extensions
8 Specification. Minor clarifications may be eligible, which could result from perceptions made in the
9 implementation and/or rollout process. These will be published in future editions of this document.

10 The latest edition always will be available at www.osci.eu/transport/OSCI2/specification.

11 Editor of this document:

12 Jörg Apitzsch, bos bremen online services GmbH & Co. Kg, ja@bos-bremen.de

13 Quality Assurance:

14 Thilo Schuster, cit GmbH, thilo.schuster@cit.de

15 Further contributors are listed in [Appendix D.].

16 Comments and questions may be addressed to the above mentioned persons.

17 **Change History (since Edition 1)**

18

Edition	as of	Author	Changes made / Comments

19

20

Table of Contents

21	1	Introduction.....	5
22	2	Document Structure	6
23	3	Document Conventions	7
24	3.1	Notational Conventions	7
25	3.2	XML Namespaces	8
26	4	Specification Conformance	10
27	4.1	Conformance Requirements	10
28	4.2	Conformance Targets	10
29	5	SOAP Version, Transport and Fault Binding.....	12
30	6	Addressing Endpoints	13
31	6.1	Use of WS-Addressing.....	13
32	6.1.1	Endpoint Reference	13
33	6.1.2	Addressing Properties – SOAP Binding	15
34	6.2	Anonymous Access and use of WS MakeConnection.....	17
35	7	Message Security, Authentication and Authorization.....	19
36	7.1	WS Security header block.....	19
37	7.2	XML Digital Signature	19
38	7.2.1	Restrictions to WS-I Basic Security Profiling	19
39	7.2.2	Format of XML Digital Signatures used for Documents	20
40	7.3	XML Encryption.....	23
41	7.3.1	End-to-end Encryption of Content Data.....	23
42	7.3.2	Encryption Cyphersuite Restrictions.....	24
43	7.4	Security Token Types	24
44	7.5	Use of WS-Trust and SAML Token.....	25
45	7.5.1	Authentication Strongness.....	26
46	7.5.2	WS-Trust Messages	27
47	7.5.3	Issued SAML-Token Details	33
48	7.5.4	Authentication for Foreign Domain Access	35
49	7.5.5	SAML-Token for Receipt- /Notification Delivery	35
50	8	OSCI specific Extensions	39
51	8.1	Message Flow Time Stamping.....	39
52	8.2	Accessing message boxes.....	40
53	8.2.1	MsgBoxFetchRequest	40
54	8.2.2	MsgBoxStatusListRequest.....	42
55	8.2.3	MsgBoxResponse.....	45
56	8.2.4	MsgBoxGetNextRequest	48
57	8.2.5	MsgBoxCloseRequest	49
58	8.2.6	Processing Rules for MsgBoxGetNext/CloseRequest.....	51
59	8.3	Receipts	52
60	8.3.1	Demanding Receipts	52
61	8.3.2	Receipt Format and Processing	55
62	8.3.3	Fetches Notification	60
63	8.4	X.509-Token Validation on the Message Route	62
64	8.4.1	X.509-Token Container.....	63
65	8.4.2	X.509-Token Validation Results	65
66	8.4.3	Verification of XKMS Validate Result Signatures	66
67	8.5	General Processing of Custom Header Faults	66

68	9	Constituents of OSCI Message Types	67
69	9.1	osci:Request	68
70	9.2	osci:Response.....	70
71	9.3	MsgBoxFetchRequest.....	72
72	9.4	MsgBoxStatusListRequest.....	73
73	9.5	MsgBoxResponse	74
74	9.6	MsgBoxGetNextRequest.....	75
75	9.7	MsgBoxCloseRequest.....	76
76	10	Policies and Metadata of Communication Nodes and Endpoints.....	78
77	10.1	General usage of Web Service Description Language	78
78	10.1.1	WSDL and Policies for MEP synchronous point-to-point	78
79	10.1.2	WSDL and Policies for asynchronous MEPs via Message Boxes	79
80	10.2	OSCI specific Characteristics of Endpoints	79
81	10.2.1	Certificates used for Signatures and Encryption	79
82	10.2.2	Endpoint Services and Limitations	83
83	10.3	WS Addressing Metadata and WS MakeConnection	85
84	10.4	WS Reliable Messaging Policy Assertions	86
85	10.5	MTOM Policy Assertion	86
86	10.6	WS Security Profile and Policy Assertions	86
87	10.6.1	Endpoint Policy Subject Assertions	86
88	10.6.2	Message Policy Subject Assertions.....	87
89	10.6.3	Algorithm Suite Assertions.....	88
90	11	Applying End-to-end Encryption and Digital Signatures on Content Data	90
91	12	Indices.....	91
92	12.1	Tables	91
93	12.2	Pictures	91
94	12.3	OSCI specific faults	91
95	12.4	Listings.....	92
96	13	References.....	93
97	13.1	Normative.....	93
98	13.2	Informative	96
99		Appendix A. Schema	97
100		Appendix B. Example: OSCI Endpoint Metadata Instance	103
101		Appendix C. Example Signature Element	105
102		Appendix D. Acknowledgements.....	107

103 **1 Introduction**

104 The Web Services Profiling and Extensions Specification **Online Service Computer Interface Transport**
105 2.0 (OSCI 2.0) is made up of five documents:

106 (1) "OSCI-Transport 2.0 – Functional Requirements and Design Objectives"

107 (2) "OSCI-Transport 2 – Technical Features Overview"

108 (3) "OSCI Transport 2.0 – General Architecture"

109 and

110 (4) "OSCI Transport 2.0 – Web Services Profiling and Extensions Specification".

111 These for documents are accomplished by a common comprehensive glossary:

112 (5) "OSCI Transport 2 – Glossary".

113 While the technical overwie and the specification and profiling documents are presented in English
114 language only, the other mentioned documents initially are available in German language¹.

115 The background and principles of the **Online Service Computer Interface (OSCI) Tranport spefiction** is
116 explained in the document "OSCI-Transport 2 – Technical Features Overview", which should be read
117 first to obtain a base understanding for the profiling and specifications outlined in the here presented
118 document.

¹ While the here presented document is under composition, the English translations of the document „OSCI Transport 2.0 – Generelle Architektur“ (general Architecture) is still outstanding.

119 **2 Document Structure**

120 Chapter [3] clarifies formal appointments concerning notational conventions, which is followed by a
121 summary of conformance targets and requirements.

122 Due to the proliferation of differing platforms and technologies in the eGovernment, it is essential to
123 ensure the different Web Service implementations are interoperable, regardless of the underlying
124 implementation and operation technology. Therefore, we mainly rely on the work which is done by the
125 Web Services Interoperability Organization², where profilings are compiled of the major Web Services
126 specifications under the aspect of best practices for Web Services interoperability. If needed to satisfy
127 the underlying OSCI requirements, we define further restrictions and processing rules in addition to
128 these profilings. This is outlined in the chapters [5 thru 7].

129 As OSCI Transport has to serve special requirements, which are not yet satisfied by currently
130 available Web Services specifications, chapter [8] specifies extensions to the WS-Stack for these
131 purposes.

132 Chapter [9] summarizes the constituent of the different OSCI message types, completed by hints
133 concerning policies and metadata definitions for nodes and endpoints of OSCI-based communication
134 networks in chapter [10].

135 Finally, in chapter [11] hints are given to realize services, which may be needed by applications for
136 end-to-end encryption and digital signature services on the content data level. Although a transport
137 protocol should be agnostic to payload carried, these services are needed to satisfy confidentiality,
138 legal bindings and non repudiation requirements.

139 In a continuing refinement process, this specification will be amended by example policies and
140 realization hints for selected classes of communication scenarios.

² see <http://www.ws-i.org/>

141 3 Document Conventions

142 3.1 Notational Conventions

143 The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD
144 NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as
145 described in [RFC2119].

146 This specification uses the following syntax to define normative outlines for messages:

- 147 • The syntax appears as an XML instance, but values in italics indicate data types instead of
148 values.
- 149 • Characters are appended to elements and attributes to indicate cardinality:
 - 150 ○ "?" (0 or 1)
 - 151 ○ "*" (0 or more)
 - 152 ○ "+" (1 or more)
- 153 • The character "|" is used to indicate a choice between alternatives.
- 154 • The characters "(" and ")" are used to indicate that contained items are to be treated as a
155 group with respect to cardinality or choice.
- 156 • An ellipsis (i.e. "...") indicates a point of extensibility that allows other child or attributes content
157 specified in this document. Additional children elements and/or attributes MAY be added at the
158 indicated extension points but they MUST NOT contradict the semantics of the parent and/or
159 owner, respectively. If an extension is not recognized it SHOULD be ignored.
- 160 • XML namespace prefixes (see section 3.2) are used to indicate the namespace of the element
161 being defined.

162 Elements and Attributes defined by this specification are referred to in the text of this document using
163 [XPath 1.0] expressions. Extensibility points are referred to using an extended version of this syntax:

- 164 • An element extensibility point is referred to using {any} in place of the element name. This
165 indicates that any element name can be used, from any namespace other than the osci:
166 namespace.
- 167 • An attribute extensibility point is referred to using @{any} in place of the attribute name. This
168 indicates that any attribute name from any namespace can be used.

169 For those parts of this specification where referenced specifications are profiled, normative statements
170 of requirements are presented in the following manner:

171 **Rnnnn** - *Statement text here*

172 where "nnnn" is replaced by a number that is unique among the requirements in this specification,
173 thereby forming a unique requirement identifier.

174 The terms "header" and "body" used in this document are used as abbreviation of "SOAP header"
175 respective "SOAP body".

176 Following legend applies for the message diagrams in this document:

- | |
|---|
| 177 • Mandatory constituents have continuous lines, optional ones are marked dashed. |
| 178 • Arrows on the left diagram side mark transport encryption requirements, those on the right
179 transport signature requirements. |
| 180 • Encrypted message parts are marked by a hatched background. |

181

182 For explanation of used abbreviations and terms see the additional document "OSCI Transport 2.0 –
183 Glossary".

184 3.2 XML Namespaces

185 Following XML namespaces are referenced:

Prefix	XML Namespace	Specification
ds	http://www.w3.org/2000/09/xmldsig#	[XMLDSIG]
dss	urn:oasis:names:tc:dss:1.0:core:schema	[DSS]
fimac	urn:de:egov:names:fim:1.0:authenticationcontext ³	[SAFE]
osci	http://www.osci.eu/ws/2008/05/transport	This document
s12	http://www.w3.org/2003/05/soap-envelope	[SOAP12]
samlac	urn:oasis:names:tc:SAML:2.0:ac	[SAMLAC]
saml2	urn:oasis:names:tc:SAML:2.0:assertion	[SAML2]
wsa	http://www.w3.org/2005/08/addressing	[WSA]
wsaw	http://www.w3.org/2006/05/addressing/wsdl	[WSAW]
wsdli	http://www.w3.org/ns/wsdl-instance	[WSDL20]
wsdl11	http://schemas.xmlsoap.org/wsdl/	[WSDL11]
wsmc	http://docs.oasis-open.org/ws-rx/wsmc/200702	[WSMC]
wsp	http://www.w3.org/ns/ws-policy	[WSPF], [WSPA]
wspmtom	http://docs.oasis-open.org/ws-rx/wsrmp/200702	[MTOMP]
wstrm	http://docs.oasis-open.org/ws-rx/wstrm/200702	[WSTRM]
wstrmp	http://docs.oasis-open.org/ws-rx/wstrmp/200702	[WSTRMP]
wsse	http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-1.1.xsd	[WSS]
wssp	http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702	[WSSP]
wsu	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd	[WSS10]
wst	http://docs.oasis-open.org/ws-sx/ws-trust/200512	[WST]
xenc	http://www.w3.org/2001/04/xmlenc#	[XENC]

³ Preliminary namespace for a SAML AuthnContext extension; proposal subject to standardization in Germany

xkms	http://www.w3.org/2002/03/xkms#	[XKMS]
xkmsEU	http://www.lsp.eu/2009/04/xkmsExt#	[XKMSEU]
xs	http://www.w3.org/2001/XMLSchema	[XMLSchema]

186 Table 1: Referenced Namespaces

187 4 Specification Conformance

188 4.1 Conformance Requirements

189 An implementation is not conformant with this specification if it fails to satisfy one or more of the
190 MUST, MUST NOT or REQUIRED level requirements defined herein.

191 A SOAP node MUST NOT use following XML namespace identifiers for the custom SOAP headers
192 defined in this specification within SOAP envelopes unless it is conformant with this specification:

- 193 • <http://www.osci.eu/ws/2008/05/osci-transport.xsd>
- 194 • <http://www.w3.org/2002/03/xkms#>
- 195 • http://www.osci.eu/2008/05/common/xkiss_ext_de.xsd.

196 Normative text within this specification takes precedence over normative outlines, which in turn take
197 precedence over the [XMLSchema] descriptions.

198 4.2 Conformance Targets

199 Conformance targets identify what artefacts (e.g., SOAP message, WSDL description, security token)
200 or parties (e.g., SOAP processor, end user) requirements apply to.

201 This allows for the definition of conformance in different contexts, to assure unambiguous
202 interpretation of the applicability of requirements, and to allow conformance testing of artefacts (e.g.,
203 SOAP messages and WSDL descriptions) and the behaviour of various parties to a Web Service (e.g.,
204 clients and service instances).

205 Requirements' conformance targets are physical artefacts wherever possible, to simplify testing and
206 avoid ambiguity.

207 The following conformance targets are used in this specification:

208 **OSCI MESSAGE** - protocol elements that profiles the SOAP Envelope, whereby following special
209 OSCI message types are defined:

210 **osci:Request, osci:Response, MsgBoxFetchRequest, MsgBoxResponse,**
211 **MsgBoxStatusListRequest, MsgBoxGetNextRequest, MsgBoxCloseRequest**

212 **OSCI GATEWAY** – an assembly of functionalities realized in software able to produce, send, receive
213 and consume OSCI Messages, hereby not concerned with SOAP Body entries (OSCI Messages for
214 MsgBox access and faults transmitted in the SOAP body excepted)

215 **DESCRIPTION** - descriptions of types, messages, interfaces and their concrete protocol and data
216 format bindings, and the network access points associated with Web Services (e.g., WSDL
217 descriptions)

218 **INITIATOR** – end point instance that generates a message according to the protocol associated with it
219 and that sends it to a RECIPIENT or MsgBox potentially through a message path that involves one or
220 multiple INTERMEDIARY(ies);

221 **RECIPIENT** – end point instance that consumes a message according to the protocol associated with
222 it.

223 **INTERMEDIARY** – node instance in the message path to the RECIPIENT which offers surplus to the
224 MESSAGE according to the protocol associated with it.

225 **MSG-BOX SERVICE** (short **MsgBox**) – specialized INTERMEDIARY instance that is able to relay
226 messages until they are pulled by the intended RECIPIENT according to the protocol defined here.

- 227 **ENDPOINT** – collective term for INITIATOR, RECIPIENT and MsgBox. Each ENDPOINT may be in
228 the role of a Security Token Requestor (STR)
- 229 **STR** – Security Token Requestor as defined by WS-Trust.
- 230 **STS** – Security Token Service as defined by WS-Trust.
- 231 **SAML-TOKEN** – Security Token as defined by SAML.

232 5 SOAP Version, Transport and Fault Binding

- 233 **R0010** - **OSCI Nodes** MUST support SOAP Version 1.2 according to [SOAP12] and constraints
 234 specified in [WSI-Basic], chapter 3 Messaging with restriction **R0020**.
- 235 **R0020** - Transport binding is restricted to HTTP/1.1, which has performance advantages and is
 236 more clearly specified than HTTP/1.0. R1140 of [WSI-Basic] (A MESSAGE SHOULD be
 237 sent using HTTP/1.1) – is superseded: A **MESSAGE** MUST be sent using HTTP/1.1.
- 238 Note that this requirement does not prohibit the use of HTTPS.
- 239 **R0030** - Errors use the SOAP fault mechanisms. The SOAP fault block according to [SOAP12]
 240 MUST be used to report information about errors occurring while processing a
 241 SOAP/OSCI message. The `s12:Fault` element MUST be carried in the SOAP body
 242 block of the network backchannel SOAP response message or – if no backchannel
 243 available in asynchronous scenarios – in the SOAP body block of a distinct message of
 244 `osci:Request`.

245 As specifications incorporated here in general define their own fault handling, this document only
 246 outlines additional fault situations specific to OSCI Transport.

247 Following information for the sub elements `s12:Fault` is supplied per fault described in this
 248 document:

249	Sub Element	Property Label	Possible values
250	<code>/Fault/Code/Value</code>	[Code]	Sender Receiver
251	<code>/Fault/Code/Value/Subcode</code>	[Subcode]	A local QName assigned to the fault
252	<code>/Fault/Reason/Text</code>	[Reason]	The English language reason explanation
253	In the fault message itself, the [Code] value MUST have a prefix of <code>s12:</code> ; the [Subcode] value prefix		
254	MUST be <code>osci:</code> .		
255	It should be noted that implementations MAY provide second-level details fields, but they should be		
256	careful not to introduce security vulnerabilities when doing so (e.g. by providing too detailed		
257	information).		

258 6 Addressing Endpoints

259 The use of WS-Addressing with SOAP 1.2-binding and WS-Addressing Metadata is mandatory for
260 OSCI Transport.

261 **R0100** - **OSCI Nodes** MUST support WS-Addressing and WS-Addressing Metadata according to
262 [WSA] and [WSAM]. Constraints apply specified in [WSI-Basic], chapter 3.6 "Support for
263 WS-Addressing Messaging" and chapter 3.7 "Use of WS-Addressing MAPs".

264 **R0110** - **OSCI Nodes** MUST support WS-Addressing SOAP Binding according to [WSASOAP],
265 whereby only the rules for binding to SOAP 1.2 apply.

266 6.1 Use of WS-Addressing

267 The use of mechanisms specified by WS-Addressing [WSA] is REQUIRED. The use of WS-
268 Addressing MUST be expressed in the syntax defined by WS-Addressing metadata [WSAM] in the
269 WSDL describing and endpoint (see chapter [10]).

270 6.1.1 Endpoint Reference

271 WS-Addressing introduces the construct of endpoint references (EPR) and defines abstract properties
272 for one-way and request-response MEPs (see [WSA] , chapter 3.1), whereas OSCI regularly uses
273 request-response MEPs. The XML Infoset representation is given in [WSA] , chapter 3.2.

274 This specification defines following restrictions on the cardinality of elements contained in a type of
275 **wsa:EndpointReference** and concretion concerning the element **wsa:ReferenceParameters** :

```
276 <wsa:EndpointReference>
277   <wsa:Address> xs:anyURI </wsa:Address>
278   <wsa:ReferenceParameters>
279     <osci:TypeOfBusinessScenario> xs:anyURI</osci:TypeOfBusinessScenario>
280   </wsa:ReferenceParameters> ?
281   <wsa:Metadata>
282     ( xmlns:wSDL="http://www.w3.org/ns/wSDL-instance"
283       wSDL:wSDLLocation= "xs:anyURI xs:anyURI"> ) |
284     xs:any *
285   </wsa:Metadata> ?
286 </wsa:EndpointReference>
```

287 **/wsa:ReferenceParameters**

288 **R0120** – If the URI value of `../wsa:Address` is not equal to
289 `"http://www.w3.org/2005/08/addressing/anonymous"`, an EPR MUST contain
290 one element **wsa:ReferenceParameters** which carries the type of business scenario
291 addressed by the message. This element is defined as type `xs:any*` and optional in
292 [WSA]. The type of business scenario MUST be tagged as URI in the OSCI namespace
293 as `osci:TypeOfBusinessScenario`.

294 Any endpoint SHOULD expose the types of business scenarios which it actually is able to
295 serve in WSDL [WSDL11] format. An XML schema definition for the Content Data to be
296 carried in the SOAP body of the message MUST be bound to the concrete tagged type of
297 business scenario. Following the WSDL binding of WS-Addressing [WSAW], each
298 **osci:TypeOfBusinessScenario** corresponds to a specific port [WSDL11] respective
299 endpoint [WSDL20].

300 Following types of business scenarios MUST be served by all OSCI endpoints:

Type of business scenario URI	Meaning
<code>http://www.osci.eu/2008/common/urn/messageTypes/Receipt</code>	Receipt type messages
<code>http://www.osci.eu/2008/common/urn/messageTypes/Notification</code>	Notification type messages
<code>http://www.osci.eu/2008/common/urn/messageTypes/Fault</code>	Fault type messages

301 Table 2: Predefined business scenario types

302 Following type of business scenerio SHOULD be served by OSCI endpoints, which are
 303 intended to be able to support a common mail-style data exchange, optional carrying any
 304 type of attachments⁴:

305 `http://www.osci.eu/2008/common/urn/messageTypes/LetterStyle`
 306 `/wsa:MetaData`

307 **R0130** - an EPR MAY have elements `/wsa:MetaData` which carry embedded or
 308 referenced metadata information assigned to this EPR.

309 Each OSCI endpoint SHOULD publish a link to its WSDL by using
 310 `wsdl:wsdlLocation`.

311 Such elements form the metadata that is relevant to the interaction with the endpoint. An
 312 Initiator MUST have knowledge about following metadata specific for OSCI about the
 313 destination he is targeting a message to, at least each OSCI endpoint SHOULD publish
 314 references to its encryption and signature certificate(s) in the OSCI specific policy
 315 `/osci:X509CertificateAssertion`⁵ by using the
 316 `wsse:SecurityTokenReference/wsse:Reference` token reference.

317 X.509-Certificate to be used for end-to-end encryption of Content Data as exposed in
 318 `/osci:X509CertificateAssertion`.

319 X.509-Certificate to possibly to be used for transport encryption (depending on concrete
 320 security policy) as exposed in `/osci:X509CertificateAssertion`.

321 X.509-Certificates used by the destination for receipt signatures, possibly those used for
 322 cryptographic time stamping, too (both exposed in
 323 `/osci:X509CertificateAssertion`).

324 Availability of qualified time stamping service and policies those apply here (as exposed in
 325 `/osci:QualTSPAssertion`⁶).

326 Possible rules applied by the destination concerning message lifetime, if messages are
 327 marked as valid for a restricted period of time (see chapter [8.1] for this issue; the
 328 endpoint behaviour id outlined in the `/osci:ObsoleteAfterAssertion`⁶ of the OSCI
 329 specific policy.

⁴ The according content data schema will be made available as addendum short after publishing of this specification

⁵ Details are defined in chapter [10.2.1]

⁶ See chapter [10.2.2]

330 These requirements and capabilities of an OSCI endpoint have SHOULD be described as
 331 policies in machine readable form; for details, see chapter [10.2]. It is advised to carry
 332 URI-references to these policies in `/wsa:MetaData`. Anyway, it is possible to embed
 333 these policies in any WSDL (fragment) describing the OSCI endpoint or even to exchange
 334 these information on informal basis out of scope of this specification.

335 6.1.2 Addressing Properties – SOAP Binding

336 This specification defines following restrictions on the cardinality of WS-Addressing message
 337 addressing properties carried as SOAP header elements as outlined in Web Services Addressing 1.0
 338 – SOAP Binding [WSASOAP]:

```

339 <wsa:To> xs:anyURI </wsa:To>
340 <wsa:ReferenceParameters>xs:any*</wsa:ReferenceParameters>
341 <wsa:From> wsa:EndpointReferenceType </wsa:From> ?
342 <wsa:ReplyTo> wsa:EndpointReferenceType </wsa:ReplyTo> ?
343 <wsa:FaultTo> wsa:EndpointReferenceType </wsa:FaultTo> ?
344 <wsa:Action>
345     http://www.osci.eu/ws/2008/05/transport/urn/messageTypes/OSCIRequest |
346     http://www.osci.eu/ws/2008/05/transport/urn/messageTypes/OSCIResponse |
347     http://www.osci.eu/ws/2008/05/transport/urn/messageTypes/MsgBoxFetchRequest
348 est |
349     http://www.osci.eu/ws/2008/05/transport/urn/messageTypes/MsgBoxStatusListRequest |
350     http://www.osci.eu/ws/2008/05/transport/urn/messageTypes/MsgBoxResponse
351 |
352     http://www.osci.eu/ws/2008/05/transport/urn/messageTypes/MsgBoxGetNextRequest |
353     www.osci.eu/ws/2008/05/transport/urn/messageTypes/MsgBoxCloseRequest
354 </wsa:Action>
355 <wsa:MessageID> xs:anyURI </wsa:MessageID>
356 <wsa:RelatesTo RelationshipType="xs:anyURI"?>xs:anyURI</wsa:RelatesTo> *
  
```

359 `/wsa:To`

360 The message final destination URI defined in `wsa:Address` is mapped to this SOAP
 361 header element which MUST be provided exactly once.

362 `/wsa:ReferenceParameters` (mapped to `osci:TypeOfBusinessScenario` in the SOAP
 363 binding)

364 **R0140** - an OSCI message MUST carry at least one element according to the SOAP
 365 mapping defined for `wsa:ReferenceParameters`. According to R0120, this is an URI
 366 carried in a SOAP header element `osci:TypeOfBusinessScenario` which is bound
 367 to the address element `wsa:To`. The SOAP header `osci:TypeOfBusinessScenario`
 368 MUST carry an attribute `wsa:IsReferenceParameter="true"`.

369 If this header element is missing or the addressed endpoint is not able to serve the
 370 concrete `osci:TypeOfBusinessScenario`, a fault MUST be generated and the
 371 message MUST be discarded:

372 Fault 1: AddrWrongTypeOfBusinessScenario

373 [Code] Sender

374 [Subcode] AddrWrongTypeOfBusinessScenario

375 [Reason] Type of Business Scenario missing or not accepted

376 `/wsa:From ?`

377 As OSCI is designed for authoritative communication, an OSCI message SHOULD carry
 378 at most one SOAP header element `wsa:From` of type `wsa:EndpointReferenceType`.
 379 If carried, the issuer of this message MUST expose here the EPR where he is able to

380 accept osci:Request messages according to R0120, R0130; it SHOULD carry the same
381 entries as `/wsa:ReplyTo`.

382 In case of an anonymous Initiator this EPR MAY contain the only child element
383 `wsa:Address` with a content of
384 "`http://www.w3.org/2005/08/addressing/anonymous`".

385 `/wsa:ReplyTo` ?

386 **R0150** - in case of non-anonymous and/or asynchronous scenarios a messages of type
387 osci:Request MUST carry exactly one SOAP header element `wsa:ReplyTo` of type
388 `wsa:EndpointReferenceType`. This MUST contain the concrete EPR of the endpoint
389 according to R0120, R0130; it denotes the final destination the Recipient MUST deliver
390 the response to. The `wsa:ReferenceParameters` of this EPR SHOULD be the same
391 as bound to the address element `wsa:To`.

392 If this element is not supplied, the osci:Response (or a fault) is delivered in the http-
393 backchannel (semantics following [WSA], anonymous URI).

394 For sake of simplification, all other OSCI message types SHOULD NOT carry this SOAP
395 header element, as for these message types reply destinations are defaulted to the
396 anonymous URI or there is no need to generate related responses at all.

397 `/wsa:FaultTo` ?

398 **R0160** - If faults related to this message shall not (or cannot in asynchronous scenarios)
399 be delivered in the network connection backchannel or it is intended to route such fault
400 messages to specialized endpoints for consuming fault messages, an OSCI message
401 SHOULD carry this optional element `wsa:FaultTo` of type
402 `wsa:EndpointReferenceType`. This MUST be a concrete EPR according to R1020,
403 R0130. To distinct such messages from other message types, the
404 `wsa:ReferenceParameters` of this EPR MUST be

```
405 <osci:TypeOfBusinessScenario>
406   www.osci.eu/2008/common/urn/messageTypes/Fault
407 </osci:TypeOfBusinessScenario>
```

408 In this case, a http response code of 500 MUST be returned in the backchannel of the
409 SOAP request and the body of the SOAP response MUST carry the fault information in
410 parallel to the fault message send to the endpoint denoted in `/wsa:FaultTo`.

411 If this element is not supplied, the fault only MUST be delivered in the http-backchannel.

412 `/wsa:Action`

413 **R0170** - this mandatory element of type `xs:anyURI` denotes the type of the OSCI
414 message and MUST carry one of the values outlined in the table below. An OSCI
415 message MUST carry exactly one `/wsa:Action` SOAP header element.

wsa:Action URIs assigned to OSCI Message Types
<code>http://www.osci.eu/ws/2008/05/transport/urn/messageTypes/ osci:Request</code>
<code>http://www.osci.eu/ws/2008/05/transport/urn/messageTypes/ osci:Response</code>
<code>http://www.osci.eu/ws/2008/05/transport/urn/messageTypes/ MsgBoxFetchRequest</code>
<code>http://www.osci.eu/ws/2008/05/transport/urn/messageTypes/ MsgBoxStatusListRequest</code>

<code>http://www.osci.eu/ws/2008/05/transport/urn/messageTypes/MsgBoxResponse</code>

<code>http://www.osci.eu/ws/2008/05/transport/urn/messageTypes/MsgBoxGetNextRequest</code>

<code>http://www.osci.eu/ws/2008/05/transport/urn/messageTypes/MsgBoxCloseRequest</code>

416 Table 3: Defined URIs for the WS Addressing Action element

417 If this header element has not one of the values outlined, the message MUST be
418 discarded and a fault MUST be generated.

419 Fault 2: **AddrWrongActionURI**

420 [Code] Sender

421 [Subcode] AddrWrongActionURI

422 [Reason] Invalid Action header URI value

423 **`/wsa:MessageID`**

424 **R0180** - this mandatory element of type **`xs:anyURI`** MUST carry a unique message ID
425 (UUID) according to IETF RFC "A Universally Unique Identifier (UUID) URN Namespace"
426 [RFC4122]. An OSCI message MUST carry exactly one **`/wsa:MessageID`** SOAP header
427 element.

428 **`/wsa:RelatesTo *`**

429 **R0190** - these optional elements of type **`xs:anyURI`** MUST be included, if a message is
430 to be seen as a response to preceding messages and in this case MUST carry the
431 **`wsa:MessageID`** SOAP header entry of those messages. This is always the case for the
432 network backchannel `osci:Response` and `MsgBoxResponse`. In case of asynchronous
433 responses on Content Data level (carried in a new `osci:Request`) the values for these
434 elements MUST be supplied by the responding Target Application. In case of an
435 `OSCI fetched Notification` (see chapter [8.3.3]), the value MUST be the one of the
436 message currently being fetched out of a `MsgBox` instance.

437 **`/wsa:RelatesTo/@RelationshipType ?`**

438 This optional attribute of type **`xs:anyURI`** SHOULD be omitted. Following the semantics
439 of [WSA], the implied value of this attribute is
440 "`http://www.w3.org/2005/08/addressing/reply`".

441 6.2 Anonymous Access and use of WS MakeConnection

442 Endpoints MAY allow anonymous access. If this is the case, it MUST be exposed in a special policy
443 assertion expressing X.509v3-Certificate token issued by public CSPs MUST be used for
444 authentication.⁷

445 In a synchronous MEP, the response can be delivered in the `http` backchannel of the request. For this
446 behaviour, WS-Addressing specifies the anonymous URI to be carried in the **`/ReplyTo`** EPR:
447 "`http://www.w3.org/2005/08/addressing/anonymous`".

⁷ An according policy template will be one part of the profilings addendum successively published from mid 2009 on

448 If the requesting Initiator has no OSCI endpoint available listening for messages (nor a MsgBox
449 service instance), there exists to EPR the response to be delivered asynchronously can be targeted
450 to.

451 For such responses to be delivered in an asynchronous way, the specification WS MakeConnection
452 [WSMC] defines mechanisms to uniquely identify anonymous endpoints as well as making responses
453 accessible for the Initiator in a response pulling manner. On the Recipient site special features have to
454 be foreseen to hold responses until they are pulled. The fact a Recipient endpoint serves (and in this
455 also requires) support of the MakeConnection protocol is indicated by a policy assertion as described
456 in chapter [10.3].

457 OSCI implementations MAY support WS MakeConnection; no profilings apply here. Special attention
458 should be taken here concerning the authentication requirements for anonymous Initiators and
459 message security to prevent unauthorized message access.

460 The mechanisms of the WS MakeConnection protocol is seen to be useful for more or less sporadic
461 OSCI based communication, where an initial registration process is not precondition to participate in
462 an OSCI network. For such use cases, example policies will be made available be one part of the
463 profilings addendum successively published from mid 2009 on.

464 7 Message Security, Authentication and Authorization

465 For the achievement of message confidentiality and integrity, the specification Web Services Security:
466 SOAP Message Security 1.1 [WSS] is incorporated. The restrictions defined in the WS-I Basic
467 Security Profile [WSI-BSP11] MUST strictly be applied by conformant implementations and MUST be
468 matched by security policies defined for OSCI endpoints and service node instances.

469 Message protection mechanisms described here by means of encrypting and digitally signing only
470 address scenarios, where potentially unsecured network connections are used for message
471 exchange. Message exchange inside closed networks may be protected by other precautions out of
472 band of this specification. But even for those scenarios it should be kept in mind that most of data and
473 identity theft attacks are driven from inside companies, administrations and other institutions.

474 Every individual endpoint and service node SHOULD expose following information by means of WS
475 Security Policies [WSSP] attached to their respective WSDL:

- 476 • Possible use of transport layer mechanisms (HTTP over SSL/TLS); if useable the profiling of
477 [WSI-BSP11], chapter 3 "Transport Layer Mechanisms" is MUST be applied⁸.
- 478 • If message layer mechanisms must be used: Which message parts have to be encrypted and
479 signed as well as security token to be used for these purposes.
- 480 • What kind of token for authentication and authorization must be provided in a message.

481 Out of band agreement on these issues between communication partners is accepted, too.

482 7.1 WS Security header block

483 No profiling going beyond WS-I Basic Security Profile [WSI-BSP11] is made to the layout and
484 semantics of the `/wsse:Security` SOAP header block as defined in Web Services Security [WSS]
485 except:

- 486 • Transport encryption and signing is achieved by means defined in [XMLDSIG] and [XENC] for
487 which profilings are made in the following subchapters [7.2] and [7.3]. As defined by security
488 policies, signature and/or encryption application to message parts is outlined in chapter [10].
- 489 • Supported security token types, outlined in chapter [7.4].

490 WS Security defines a Timestamp element for use in SOAP messages. OSCI places the following
491 constraint on its use:

- 492 **R0200** - A SOAP header `/wsse:Security` MUST contain exactly one element
493 `/wsu:Timestamp`. This supersedes R3227 of [WSI-BSP11].⁹

494 7.2 XML Digital Signature

495 7.2.1 Restrictions to WS-I Basic Security Profiling

496 The profilings of [WSI-BSP11], chapter 8 "XML-Signature" is MUST be applied with following
497 restrictions going beyond them:

- 498 **R0300** - Transform algorithm "`http://www.w3.org/2001/10/xml-exc-c14n#`" is
499 RECOMMENDED. This supersedes R5423 and R5412 of [WSI-BSP11] to clarify; this is
500 the recommended algorithm of the list of algorithms which MUST be used following [WSI-
501 BSP11].

⁸ Applicable TLS/SSL versions and cyphersuites are defined here

⁹ "MUST NOT contain more than one" is profiled by [WSI-BSP11]

502 **R0310** - As the digest algorithm SHA-1 is seen to be weak meanwhile, one of following digest
503 method algorithms MUST be used:

Digest method algorithms
http://www.w3.org/2001/04/xmlenc#sha256
http://www.w3.org/2001/04/xmlenc#sha512
http://www.w3.org/2001/04/xmlenc#rsa-ripemd160

504 Table 4: Digest method: allowed algorithm identifiers

505 The use of SHA-256 (<http://www.w3.org/2001/04/xmlenc#sha256>) as digest
506 method algorithm is RECOMMENDED. This supersedes R5420 of [WSI-BSP11]¹⁰.

507 **R0320** - As the digest algorithm SHA-1 is seen to be weak meanwhile, one of the signature
508 method algorithms listed here MUST be used:

Asymmetric signature method algorithms
http://www.w3.org/2001/04/xmldsig-more#rsa-sha256
http://www.w3.org/2001/04/xmldsig-more#rsa-sha512
http://www.w3.org/2001/04/xmldsig-more/rsa-ripemd160
Symmetric signature method algorithms
http://www.w3.org/2001/04/xmldsig-more#hmac-sha256
http://www.w3.org/2001/04/xmldsig-more#hmac-sha512

509 Table 5: Signature method: allowed algorithm identifiers

510 RECOMMENDED signature method algorithms are
511 <http://www.w3.org/2001/04/xmldsig-more#rsa-sha256> and
512 <http://www.w3.org/2001/04/xmldsig-more#hmac-sha256>. This supersedes
513 R5421 of [WSI-BSP11]¹¹.

514 **NOTE** on R0310, R0320: The URI-Values of the attributes `ds:SignatureMethod/@Algorithm`
515 and `ds:DigestMethod/@Algorithm` are fixed to identifiers resulting from the actual list of strong
516 hash algorithms published in [AlgCat]. One of the values outlined below MUST be chosen. *This*
517 *enumeration is subject to future changes, in case of one of the algorithms must be seen to get weak.*

518 7.2.2 Format of XML Digital Signatures used for Documents

519 Besides securing message integrity, digital signatures are used in OSCI Transport to sign
520 distinguished XML documents like policies and receipts, which in case of juridical conflicts must be
521 usable as proof.

522 Here, the national signature laws and ordinances must be considered; in consequence profilings of
523 relevant standards have already been derived as well as classification of applicability of cryptographic

¹⁰ SHOULD is defined by [WSI-BSP11] for <http://www.w3.org/2000/09/xmldsig#sha1>

¹¹ SHOULD is defined by [WSI-BSP11] for signature method algorithms based on SHA-1

524 algorithms. This leads to a profiling of those XML Digital Signatures, which are applied on documents
525 as advanced or qualified signatures using an appropriate X.509v3-Certificate.

526 In summary, following profiling of [XMLDSIG] and [XAdES] applies:

527 **R0400** - The detached XML Signature format MUST be used and the signed content, if part of the
528 message (child of SOAP envelope), be referenced by the local (fragment) URI mechanism
529 as defined in [RFC2396]. Referencable fragments of message parts MUST carry an
530 attribute of type **xs:ID**. The constraints of the XML 1.0 [XML 1.0] ID type MUST be met.
531 The generation of unique ID attribute value MUST follow [RFC4122], this value SHOULD
532 be concatenated to a preceding string "uuid:".

533 **R0410** - A **ds:Signature** element MUST contain at least one **ds:Object** child element to carry
534 the signing time and a reference to the certificate used for signing. The format of this child
535 element MUST conform to definitions given by [XAdES] with following restrictions applied
536 here:

537 It MUST contain exactly on child element **xades:QualifyingProperties** including
538 the mandatory child element
539 **xades:SignedProperties/xades:SignedSignatureProperties** and an optional
540 child element **xades:UnsignedProperties**, which is foreseen to carry a qualified
541 timestamp over the signature itself in the child element
542 **xades:UnsignedSignatureProperties/xades:SignatureTimeStamp**.

543 Child elements of **xades:SignedSignatureProperties** which MUST be present are
544 **xades:SigningTime** and information about the certificate used for signing in
545 **xades:SigningCertificate**.

546 **R0420** - As consequence of R0300 and R0310, a **ds:Signature** element MUST contain at least
547 two **ds:Reference** child elements for referencing at least one detached content and the
548 elements in **ds:Object** to be included in the signature calculation.

549 **R0430** - Exclusive canonicalization MUST be applied be used to address requirements resulting
550 from scenarios where subdocuments are moved between contexts. The URI-Value of the
551 attribute **ds:CanonicalizationMethod/@Algorithm** is fixed to
552 "**http://www.w3.org/2001/10/xml-exc-c14n#**".¹²

553 **R0440** - Signatures are only applicable on base of X.509v3-Certificates which MUST conform to
554 [COMPKI]. The child elements **ds:RetrievalMethod** and **ds:X509Data** of
555 **ds:KeyInfo** MUST be present. All other choices according to [XMLDSIG] for
556 **ds:KeyInfo** MUST NOT be present. In consequence, the attribute
557 **ds:RetrievalMethod/@Type** MUST carry a value of
558 "**http://www.w3.org/2000/09/xmlsig/X509Data**".

559 **R0450** - The child elements **ds:X509IssuerSerial** and **ds:X509Certificate** of
560 **ds:X509Data** MUST be present; the child element **ds:X509CRL** SHOULD NOT be
561 present to avoid significant data overload of signature elements to be expected in case of
562 including CRLs. All other choices according to [XMLDSIG] for **ds:X509CRL** and
563 **ds:X509SKI** MUST NOT be present.

564 Details profiling and restrictions are defined by the following normative outline:

```
565 <ds:Signature Id="xs:ID">
566   <ds:SignedInfo Id="xs:ID" ?>
567     <ds:CanonicalizationMethod
568       Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"
569     </ds:CanonicalizationMethod>
```

¹² See also: R5404 of [WSI-BSP11]

```

570
571     <ds:SignatureMethod Algorithm=
572         http://www.w3.org/2001/04/xmldsig-more#rsa-sha256 |
573         http://www.w3.org/2001/04/xmldsig-more#rsa-sha512 |
574         http://www.w3.org/2001/04/xmldsig-more/rsa-ripemd160
575     </ds:SignatureMethod>
576
577     <ds:Reference Id="xs:ID" ?
578         Type="http://uri.etsi.org/011903/v1.1.1/#SignedProperties"
579         URI="xs:anyURI">
580     <ds:Transforms/> ?
581         <ds:DigestMethod Algorithm=
582             http://www.w3.org/2001/04/xmlenc#sha256 |
583             http://www.w3.org/2001/04/xmlenc#sha512 |
584             http://www.w3.org/2001/04/xmlenc#rsa-ripemd160
585         </ds:DigestMethod>
586     <ds:DigestValue> xs:base64Binary </DigestValue>
587     <ds:Reference>
588
589     ( <ds:Reference Id="xs:ID" ?
590         Type="xs:anyURI"
591         URI="xs:anyURI">
592     <ds:Transforms/> ?
593         <ds:DigestMethod Algorithm=
594             http://www.w3.org/2001/04/xmldsig-more#rsa-sha256 |
595             http://www.w3.org/2001/04/xmldsig-more#rsa-sha512 |
596             http://www.w3.org/2001/04/xmldsig-more/rsa-ripemd160
597         </ds:DigestMethod>
598         <ds:DigestValue> xs:base64Binary </DigestValue>
599     </ds:Reference> ) +
600
601 </ds:SignedInfo>
602
603 <ds:SignatureValue Id="xs:ID" ?> xs:base64Binary </ds:SignatureValue>
604
605 <ds:KeyInfo Id="xs:ID" ?>
606     <ds:RetrievalMethod
607         Type="http://www.w3.org/2000/09/xmldsig/X509Data"/>
608     <ds:X509Data>
609         <ds:X509IssuerSerial>
610             <ds:X509IssuerName> xs:string </ds:X509IssuerName>
611             <ds:X509SerialNumber> xs:integer </ds:X509SerialNumber>
612         </ds:X509IssuerSerial>
613         <ds:X509Certificate> xs:base64Binary </ds:X509Certificate>
614         <ds:X509CRL/> ?
615     </ds:X509Data>
616 </ds:KeyInfo>
617
618 <ds:Object Id="xs:ID">
619     <xades:QualifyingProperties Target="...">
620         <xades:SignedProperties>
621             <xades:SignedSignatureProperties Id="xs:ID">
622                 <xades:SigningTime> xs:dateTime </xades:SigningTime>
623                 <xades:SigningCertificate>
624                     <xades:Cert>
625                         <xades:CertDigest>
626                             <ds:DigestMethod> Algorithm=
627                                 http://www.w3.org/2001/04/xmldsig-more#rsa-sha256 |
628                                 http://www.w3.org/2001/04/xmldsig-more#rsa-sha512 |

```

```

629         http://www.w3.org/2001/04/xmldsig-more/rsa-ripemd160
630     </ds:DigestMethod>
631     <ds:DigestValue> xs:base64Binary </DigestValue>
632 </xades:CertDigest>
633 <xades:IssuerSerial>
634     <ds:X509IssuerName> xs:string </ds:X509IssuerName>
635     <ds:X509SerialNumber>
636         xs:integer
637     </ds:X509SerialNumber>
638 </xades:IssuerSerial>
639 </xades:Cert>
640 </xades:SigningCertificate>
641 </xades:SignedSignatureProperties>
642 </xades:SignedProperties>
643
644 ( <xades:UnsignedProperties Id="xs:ID" ?>
645     <xades:UnsignedSignatureProperties id="xs:id" ?>
646         <xades:SignatureTimeStamp id="xs:ID" ?>
647             ( <ds:CanocalizationMethod
648                 Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"
649             </ds:CanocalizationMethod> ) ?
650         <xades:EncapsulatedTimeStamp>
651             Id="xs:ID" ? Encoding="xs:ID" ?>
652                 xs:base64Binary
653         </xades:EncapsulatedTimeStamp>
654         </xades:SignatureTimeStamp>
655     </xades:UnsignedSignatureProperties>
656 </xades:UnsignedProperties> ) ?
657
658 </xades:QualifyingProperties>
659 </ds:Object>
660 <ds:Object Id="xs:Id" ?/> ?
661 </ds:Signature>

```

662 The outline above shows mandatory and optional elements and their cardinality restrictions. For a
663 detailed description of elements and attributes in the outline above, see [XMLDSIG] and [XAdES].

664 For illustration, an example is given for an instance of such a signature element in [0].

665 7.3 XML Encryption

666 In general, the profilings of [WSI-BSP11], chapter 9 "XML Encryption" is MUST be applied. If
667 encryption is applied, the SOAP envelope, header, or body elements MUST NOT be encrypted.
668 Encrypting these elements would break the SOAP processing model and is therefore prohibited (see
669 R5607 of [WSI-BSP11]).

670 Restrictions going beyond [WSI-BSP11] are defined in the following subchapters.

671 7.3.1 End-to-end Encryption of Content Data

672 Following general rules apply in addition to those presented in chapter [7.3.2]:

673 **R0400** - A SOAP message body block MUST be encrypted for the intended Ultimate Recipient
674 following [XENC] using the public key of his X.509v3 encryption certificate.

675 **R0410** - A hybrid encryption algorithm MUST be applied: First a random session key is generated
676 for a symmetric encryption algorithm. Using this key, the SOAP body blocks are
677 encrypted. In a second step the session key is encrypted with the public encryption key of
678 the Ultimate Recipient. The encrypted data and the encrypted session key build up the
679 resulting SOAP body block of the message.

- 680 **R0420** - It MUST be ensured that not the same session key is used for data that are directed to
681 different Ultimate Recipients.

682 7.3.2 Encryption Cyphersuite Restrictions

- 683 **R0500** - One of following symmetric block encryption algorithms MUST be used:

Encryption Algorithm	Algorithm Identifier
Two-Key-Triple-DES	http://www.w3.org/2001/04/xmlenc#tripleDES-cbc
AES-128	http://www.w3.org/2001/04/xmlenc#aes128-cbc
AES-192	http://www.w3.org/2001/04/xmlenc#aes192-cbc
AES-256	http://www.w3.org/2001/04/xmlenc#aes256-cbc

684 Table 6: Symmetric encryption algorithms

- 685 **R0510** - Encryption of symmetric keys MUST be performed by means of RSAES-PKCS1-v1_5
686 [PKCS#1]. The value of `xenc:EncryptionMethod` MUST be

687 `"http://www.w3.org/2001/04/xmlenc#rsa-1_5"`

- 688 **R0520** - The modulus length of a RSA key pair has to be at least 2048 bit.

689 7.4 Security Token Types

690 To be extensible, the WS-Security specification has not bound to specific security token types. For this
691 version of OSCI Transport, token types outlined in following table MAY be used for authentication,
692 message signature and encryption operations. Profilings of those token types have been specified by
693 the OASIS Web Services Security Technical Committee.

Security Token Type	Support	Value of <code>wsse:BinarySecurityToken/@ValueType</code> and <code>wsse:SecurityTokenReference/wsse:KeyIdentifier/@ValueType</code>	Profiling Reference
SAMLV2.0	MUST	http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0	[WSSSAML]
X.509v3-Certificate	MUST	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3	[WSSX509]
Kerberos	MAY	http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#GSS_Kerberosv5_AP_REQ	[WSSKERB]
Username	MAY	Defined in WS-Security as <code>wsse:UsernameToken</code>	[WSSUSER]

694 Table 7: Security token types – support requirements

- 695 **R0600** - SAMLV20-Token MUST be used for authentication and message security within Trust
696 Domains as well as for cross domain message exchange – except R0610 applies.

697 If access of anonymous Initiators shall be supported, Public Key Infrastructure MUST be used
698 applying X.509v3-Certificates:

699 **R0610** - X.509v3-Certificate token issued by CAs MUST be used for authentication and message
700 security for scenarios allowing anonymous access. Validity of used certificates MUST be
701 verifiable by means of OCSP, LDAP or CRL.

702 The node a message is targeted to MUST verify the certificate validity; in case a value
703 other than valid at time of usage is stated, the message MUST be discarded and a fault
704 MUST be generated.

Fault 3: AuthnCertNotValid

[Code] Sender

[Subcode] AuthnCertNotValid

[Reason] Authentication certificate not stated to be valid

705
706
707
708
709 More information about the certificate validation results SHOULD be provided in the fault
710 [Details] property in this case. It is strongly RECOMMENDED to log such faults to be able
711 to detect possible security violation attacks.

712 **R0620** - X.509v3-Certificates used for authentication MUST have set the key usage extension set
713 to "digitalSignature". If the "nonRepudiation" key usage is set, these certificates MUST
714 not be used for authentication.¹³

715 Context conformant usage of certificates and their validity SHOULD be controlled by STS
716 respective message initiating instances to avoid subsequent violations of this requirement.
717 The node a message is targeted to MUST verify conformance this requirement; in case of
718 wrong key usage set, the message MUST be discarded and a fault MUST be generated.

Fault 4: AuthnCertInvalidKeyUsage

[Code] Sender

[Subcode] AuthnCertInvalidKeyUsage

[Reason] Certificate not permitted for authentication

719
720
721
722
723 Token of type Username and Kerberos MAY be used for authentication and securing messages inside
724 closed communication domains, where security and trust is given by means out of band of this
725 specification.

726 7.5 Use of WS-Trust and SAML Token

727 In general, means of WS-Trust¹⁴ SHOULD be used where all communication partners of a Trust
728 Domain are registered at an IdP, having a STS available for issuing SAML-Token.

729 **R0630** - Each access to an endpoint MUST be authorized by a STS instance of the endpoints
730 Trust Domain. A STS MUST be able to confirm the Requestors identity on base of
731 presented credentials.

732 For a given Trust Domain, the definition of a standard security policy and SAM Token layout is
733 RECOMMENDED, which can basically be used for message exchange inside this domain. If certain
734 services have special authentication and/or authorization requirements, this can be propagated in
735 according security policies bound to these services respective endpoints.

¹³ The signature used for authentication must not be confused with the legal declaration of intent given by a (qualified) digital signature.

¹⁴ An overview for use of WS-Trust and SAML-Token is given in chapter **[Fehler! Verweisquelle konnte nicht gefunden werden.]**.

736 7.5.1 Authentication Strongness

737 Access authorization at least is given by the assurance of a certain level of authentication of the STR.
738 Trustworthiness of the STR identity confirmation thru an STS is given by the strongness of following
739 two processes:

- 740 • Initial registration of an endpoint at his IdP – organizational rules that applied for the degree of
741 trustworthiness initial subject identification
- 742 • Mechanisms used for authentication at the time of requesting identity confirmation from the
743 STS to match claimed and conformed identity.

744 [SAML2] and [SAMLAC] specify an authentication statement `saml:AuthnStatement` to carry such
745 information. Differentiated authentication context details may be included herein. To simplify
746 processing and interoperability, following ascending levels for strongness of registration and
747 authentication are defined¹⁵:

- 748 • `urn:de:egov:names:fim:1.0:securitylevel:normal`
- 749 • `urn:de:egov:names:fim:1.0:securitylevel:high`
- 750 • `urn:de:egov:names:fim:1.0:securitylevel:veryhigh`

751 Each level matches operational rules which must be defined, published and continuously maintained
752 by appropriate institutions, i.e. government agencies concerned to data protection.¹⁶

753 [SAFE] defines extensions to the SAML authentication context element to carry the levels of
754 registration and authentication as follows:

```
755 <samlac:Extension>
756   <fimac:SecurityLevel>
757     <fimac:Authentication>
758       urn:de:egov:names:fim:1.0:securitylevel:normal |
759       urn:de:egov:names:fim:1.0:securitylevel:high |
760       urn:de:egov:names:fim:1.0:securitylevel:veryhigh |
761     </fimac:Authentication> ?
762     <fimac:Registration>
763       urn:de:egov:names:fim:1.0:securitylevel:normal |
764       urn:de:egov:names:fim:1.0:securitylevel:high |
765       urn:de:egov:names:fim:1.0:securitylevel:veryhigh |
766     </fimac:Registration> ?
767   </fimac:SecurityLevel> ?
768 </samlac:Extension> ?
```

769 This outline is preliminary to be seen as normative.

770 `/samlac:Extension ?`

771 Optional container carrying the extension; to be included in a SAML assertion in the
772 `samlac:AuthenticationContextDeclaration` element.

773 `.../fimac:SecurityLevel ?`

774 Optional container carrying the detail elements.

775 `.../fimac:SecurityLevel/fimac:Authentication ?`

776 Optional authentication level statement of type restriction to `xs:anyURI`; if present, the
777 URI value MUST be one of the enumerations listed above.

778 `.../fimac:SecurityLevel/fimac:Registration ?`

¹⁵ Preliminary URIs proposed by the SAFE-Project; subject to standardization activities by German administration

¹⁶ Definition of such rules can not be a matter of this specification. An example for a level “veryhigh” could be a registration data confirmation on base of presenting Id Cards and subsequent authentication using authentication certificates issued by accredited CAs.

779 Optional registration strongness statement of type restriction to **xs:anyURI**; if present,
780 the URI value **MUST** be one of the enumerations listed above.

781 If a SAML token of a message addressed to an endpoint does not match the minimal security level
782 requirements of this endpoint, the message **MUST** be discarded and a fault **MUST** be generated.

783 **Fault 5: AuthnSecurityLevelInsufficient**

784 [Code] Sender

785 [Subcode] AuthnSecurityLevelInsufficient

786 [Reason] Insufficient strongness of authentication or registration

787 Detailed information on the security level requirements **SHOULD** be provided in the fault [Details]
788 property in this case.

789 To facilitate the acquisition of an appropriate SAML token for the Initiator, endpoints **SHOULD**
790 describe there requirements on authentication strongness by means of WS-Policy as outlined in
791 chapter [].

792 **7.5.2 WS-Trust Messages**

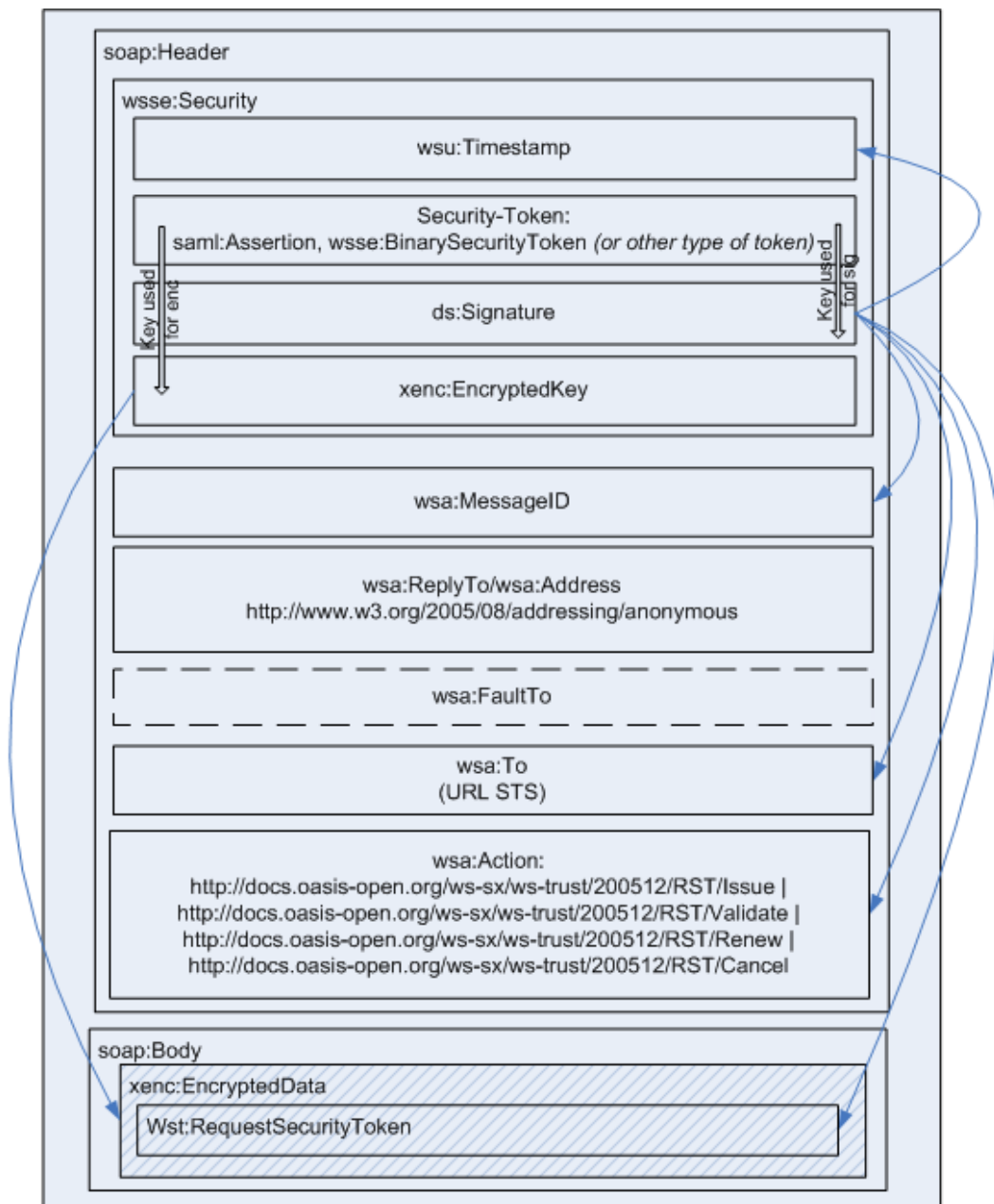
793 Conformant OSCI Gateway implementations **MUST** support the SOAP message types and bindings
794 defined by WS-Trust:

- 795 • Issue
- 796 • Validate
- 797 • Cancel.

798 The WS-Trust Renew-Binding **SHOULD** be supported for convenience; this functionality is supplied
799 by most STS-Implementations.

800 For clarification, an overview is given in following subchapters to the constituents of these message
801 types. For the exact definition of the according XML Infoset see [WST]; the present document
802 concentrates on restrictions to be applied by OSCI conformant implementations and a few hints.

803 **7.5.2.1 Request Security Token (RST)**



804
805 Figure 1: Request Security Token Message

806 SOAP header blocks:

807 **/wsse:Security**

808 This header block MUST be present, carrying message protection data and Initiator
809 authentication information according the security policy of the STS the RST message is
810 targeted to.

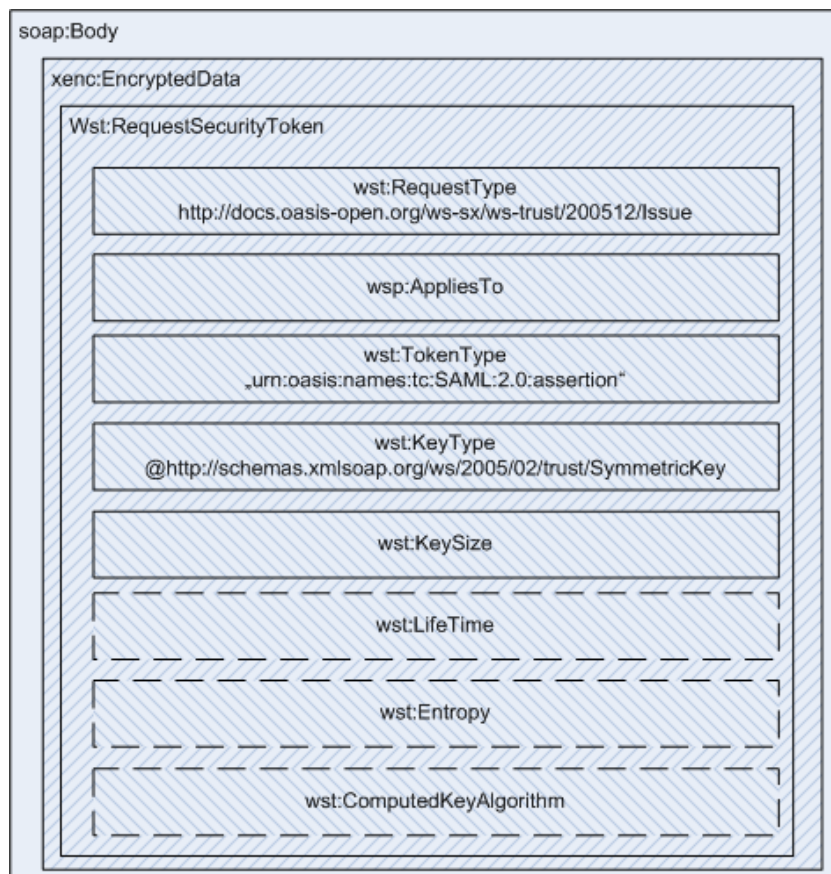
811 **/wsse:Security/wsue:Timestamp**

812 According to R0200, this header block MUST be present.

813 **/wsse:Security/[Security-Token]**

814 Security tokens MUST be used for signing and encrypting message parts. **ds:KeyInfo**
815 elements of subsequent **ds:Signature** or **xenc:EncryptedKey** elements MAY point
816 to security tokens carried here.

- 817 [Table 7] lists the security token types which MUST or MAY be supported.
- 818 Security tokens MUST be embedded or referenced. Referenced tokens MUST be
819 dereferencable by the targeted STS.
- 820 The Requestors security token MUST be used for signing the above marked message
821 parts.
- 822 **/wsse:Security/ds:Signature**
- 823 A signature containing **ds:Reference** elements to all message parts marked above to
824 be included in the signature.
- 825 **/wsse:Security/xenc:EncryptedKey**
- 826 The RST contained in the SOAP body block MUST be encrypted for the targeted STS.
827 This is a symmetric key which MUST be encrypted with the public key of the STS X.509v3
828 encryption certificate. Rules outlined in chapter [7.3] apply. It is assumed, that the STS
829 encryptions certificate is made available to all endpoints inside the STS Trust Domain out
830 of band of this specification.
- 831 **/wsa:***
- 832 All WS-Addressing headers MUST (if in continuous blocks) /MAY (if in dashed blocks) be
833 supplied by the Requestor.
- 834 **/wsa:Action**
- 835 Depending on the type of WS-Trust request, one of the URIs outlined above MUST be
836 supplied. This URI MUST be adequate to the respective body block content.
- 837 The SOAP body block MUST conform to the definitions of WS-Trust, whereby following restrictions
838 and recommendations apply for the WS-Trust Issue request type.



839

840

Figure 2: Request Security Token, Body for Issue Request

841 **/wsp:AppliesTo**

842 This element MUST be present; the value assigns a domain expression for the desired
843 application scope of the SAML-Token.

844 NOTE: For easy of message exchange inside a Trust Domain, it is RECOMMENDED to
845 choose an expression (i.e. URL pattern) accepted at least by a MsgBox instance for all
846 Recipients nodes using this MsgBox instance. This leverages the burden and overhead,
847 which would be given by a **/wsp:AppliesTo** value assignment to a concrete Recipient
848 EPR.

849 **/wst:TokenType**

850 **R0700:** This element MUST be present; the value is restricted to SAML V2.0 assertion
851 type:

852 `urn:oasis:names:tc:SAML:2.0:assertion`

853 **/wst:KeyType**

854 **R0710:** This element MUST be present; the value is restricted to:

855 `http://docs.oasis-open.org/ws-sx/ws-trust/200512/SymmetricKey`

856 **/wst:KeySize**

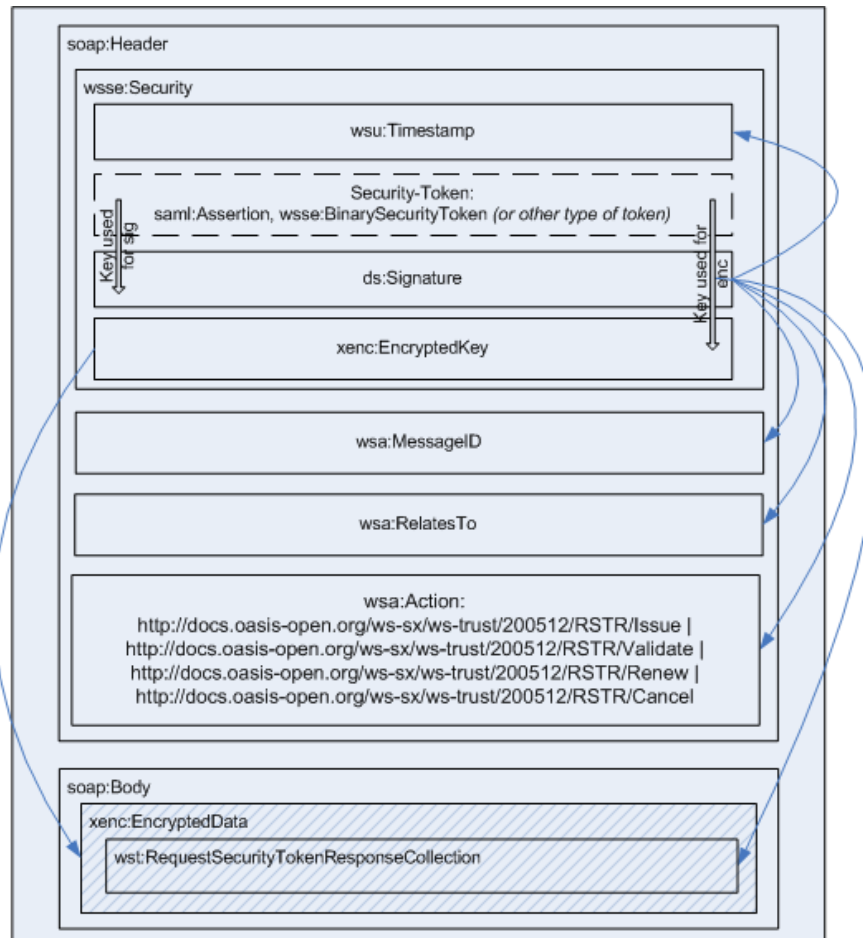
857 **R0720:** This element MUST be present; the key size MUST be greater or equal 256 Bit.

858 Use and values of elements marked optional are subject to used STS instance specific policies.

859 Recommendations will we given as part of the amendments to be worked out for this specification in
860 2009 ff.

861 **7.5.2.2 Request Security Token Response (RSTR)**

862 The SOAP header resembles the one of the RST message:



863

864 Figure 3: Request Security Token Response Message

865

865 Differences to the RST message:

866

866 **/wsse:Security/xenc:EncryptedKey**

867

867 The RSTRC contained in the SOAP body block MUST be encrypted for the token
 868 Requestor. This is a symmetric key which MUST be encrypted with the public key of the
 869 Requestors X.509v3 encryption certificate. Rules outlined in chapter [7.3] apply.

870 **/wsa:***

871

871 All WS-Addressing headers MUST (if in continuous blocks) /MAY (if in dashed blocks) be
 872 supplied by the STS.

872

873 **/wsa:Action**

874

874 Depending on the type of WS-Trust response, one of the URIs outlined above MUST be
 875 supplied. This URI MUST be adequate to the respective body block content.

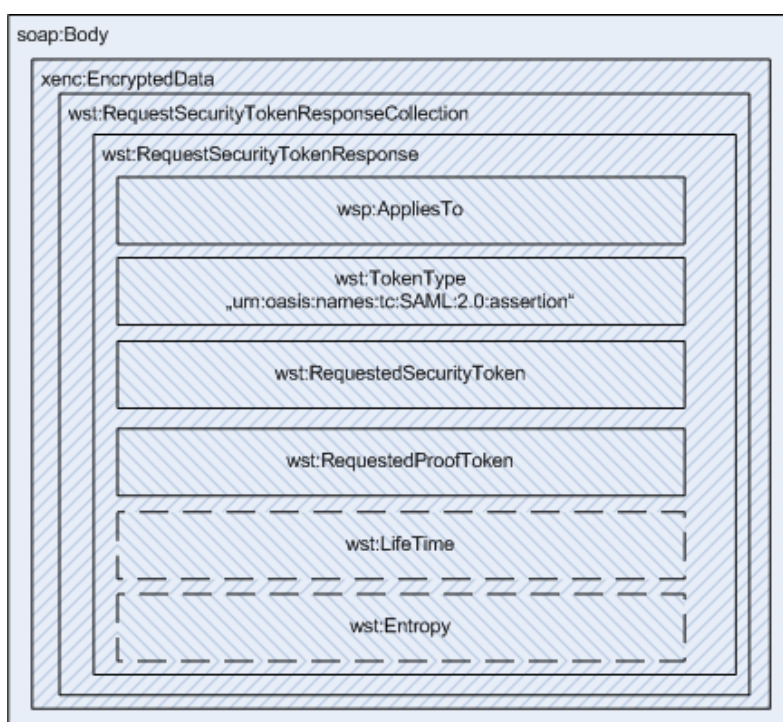
875

876

876 The decrypted SOAP body block MUST conform to the definitions of WS-Trust. No restrictions or
 877 profilings apply.

877

878 The SOAP body block MUST conform to the definitions of WS-Trust:



879

880 Figure 4: Request Security Token, Body for Issue Response

881 Short description of the constituents of `/wst:RequestSecurityTokenResponse`, which is always
882 wrapped by a `/wst:RequestSecurityTokenResponseCollection` (see [WST] for details):

883 `/wsp:AppliesTo`

884 Carries the value assignment for the desired application scope of the requested security
885 token – copied from the according request element.

886 `/wst:TokenType`

887 Carries the token type, which MUST be the one of the according request element.

888 `/wst:RequestedSecurityToken`

889 Carries the requested SAML-Token, including a symmetric key encrypted for the endpoint
890 at which the SAML-Token is needed for authentication purposes. Details explained in
891 chapter [7.5.3].

892 `/wst:RequestedProofToken`

893 Carries information enabling the Requestor to deduce the symmetric key. In case the key
894 was generated by the STS solely, this is the key itself.

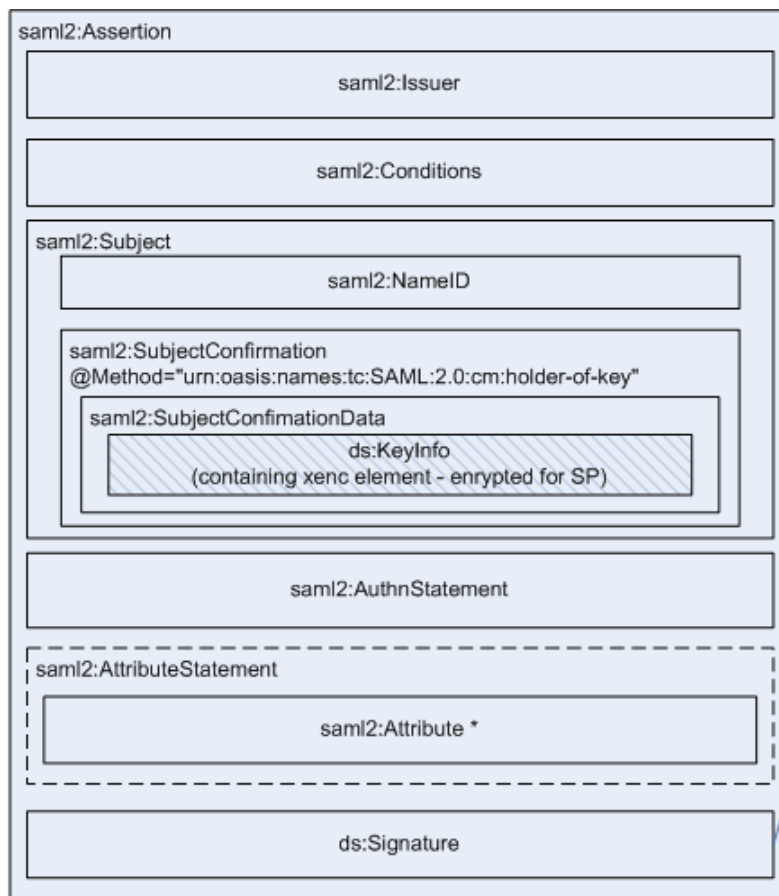
895 In case computed of two entropy values, this is the algorithm and the element

896 `/wst:Entropy ?`

897 MUST be present carrying the entropy value used by the STS for key computation.

898 `/wst:LifeTime ?`

899 Optional element carrying the validity duration period of this RSTR; SHOULD be
900 recognized by the Requestor and processed according to his needs to avoid using
901 security tokens being/getting invalid.

902 **7.5.3 Issued SAML-Token Details**

903

904 Figure 5: SAML 2.0 Assertion constituents

905 Short description of the constituents of a `/saml2:Assertion`, for XML Infoset details see [SAML2]
 906 and [SAMLAC]. The concrete token request requirements and layout of issued token at least has to be
 907 matched with the capabilities of the used STS instances. For implementations to be operated in
 908 context of the German administration it is strongly RECOMMENDED to follow requirements and
 909 recommendations given by the concept [SAFE].

910 `/saml2:Issuer`911 Attributes of the STS issuing this assertion; for details see `saml2:NameIDType`912 `/saml2:Conditions`

913 **R0730:** Detailed validity conditions element MUST be present; for details see
 914 `saml2:ConditionsType`. MUST at least outline the validity period attributes
 915 `NotBefore`, `NotOnOrAfter`.

916 `/saml2:Subject`

917 **R0740:** Presence of this element is REQUIRED. The subelements of this container
 918 provide STR identification details.

919 `/saml2:Subject/saml2:NameID`

920 Attributes of the STR; for details see `saml2:NameIDType`. It MUST at least contain a
 921 unique string identifying the STR.

922 `/saml2:Subject/saml2:SubjectConfirmation`

923 This container exposes STS information for the SP enabling it to assure the SR is the one
 924 stated in `/saml2:NameID` and authorized to use this token.

- 925 `/saml2:Subject/saml2:SubjectConfirmation/@Method`
- 926 **R0750:** Attribute outlining the confirmation method; MUST be the "holder of key"
927 confirmation method.
- 928 `/saml2:Subject/saml2:SubjectConfirmation/saml12:SubjectConfirmationData`
- 929 **R0760:** Presence of this element is REQUIRED; it exposes STS information for the SP
930 enabling it to assure the SR is the one owning key for this SAML assertion.
- 931 `/saml2:Subject/saml2:SubjectConfirmation/saml12:SubjectConfirmationData/ds:`
932 `key`
- 933 **R0770:** This element MUST carry the key in a `xenc:EncryptedKey` element. The key
934 MUST be encrypted for the SP using the public key of its X.509v3 encryption certificate,
935 which for this purpose MUST be made available to the STS.
- 936 NOTE on the endpoint encryption certificate the SAML token is targeted to:
- 937 The access to this certificate through the token issuing STS is of band of this specification;
938 this is a matter of Trust Domain policies and an implementation issue which MUST have
939 no effect on interoperability. No standardized mechanisms are foreseen by WS-Trust, to
940 include a certificate in a RST message for the purpose of key encryption for the SP. It is
941 strongly RECOMMENDED, to relate the `/wsp:AppliesTo` request value (which might
942 be a pattern, too – see RST body description in chapter [7.5.2.1]) to this encryption
943 certificate.
- 944 `/saml2:AuthnStatement`
- 945 **R0780:** Presence of this element is REQUIRED.
- 946 It MUST contain an element `/saml2:AuthnContext` with an attribute `@AuthnInstant`
947 outlining the time instant the authentication took place.
- 948 `/saml2:AuthnContext` MUST contain an element `/saml2:AuthnContextClassRef`
949 outlining the authentication method used by the SR.¹⁷
- 950 `/saml2:AuthnContext` MUST further contain an element
951 `/saml2:AuthnContextDecl` carrying the extensions for authentication strongness as
952 defined in chapter [7.5.1].
- 953 `/saml2:AttributeStatement ?`
- 954 Usage of attribute statements of type `saml12:AttributeType` is RECOMMENDED. In
955 many scenarios subject attributes like affiliation to certain groups or roles are used for the
956 assignment detailed rights, functions and data access. Hence attributes are specific to
957 application scenarios, their names, values and semantics are subject to the overall design
958 of a domain information model, which is not addressed by this specification.¹⁸
- 959 `/ds:Signature`
- 960 The issuing STS has to sign the whole SAML-Token.

¹⁷ See [SAMLAC] and [SAFE] for details; i.e. a X509v3 certificate from a smartcard was used for authentication, the value would be `urn:oasis:names:tc:2.0:ac:classes:SmartcardPKI`

¹⁸ Suggestions for use in German eGovernment, especial eJustice, are made in [SAFE]

961 If a SAML token does not match one or more of the formal requirements 0730-0780, the token
 962 consuming node MUST generate a fault and discard the message.

963 **Fault 6: AuthnTokenFormalMismatch**

964 [Code] Sender

965 [Subcode] AuthnTokenFormalMismatch

966 [Reason] Authentication token present does not match formal requirements.

967 More information MAY be given in the fault [Details] property, but care should be taken to introduce
 968 security vulnerabilities by providing too detailed information.

969 **7.5.4 Authentication for Foreign Domain Access**

970 To authenticate and authorize access to foreign TD endpoints, these endpoints MUST be able to
 971 validate the SAML-Token contained in the message. The specification WS-Federation 1.1 ([WSF],
 972 chapter 2.4) outlines several possible trust topologies; for simplification, two of those described below
 973 are selected to be applicable for this version of the OSCI specification.

974 A new version 1.2 of WS-Federation is about to be approved by the OASIS WSFED Technical
 975 Committee while publishing the here presented version of OSCI Transport. WS-Federation 1.2 will be
 976 incorporated in a follow-up of OSCI Transport. So far, the WS Federation metadata model is not yet
 977 been taken in account for usage in OSCI 2.0 based infrastructures.

978 Precondition for cross domain message exchange is an established trust relationship between the
 979 Initiators STS and the one of the foreign TD. This i.e. can be achieved by trust in the STS signature
 980 using its signing certificate as a trust anchor.

981 One useable trust model is, a SAML-Token issued by the foreign STS must be aquired for accessing
 982 endpoints in this TD. Authentication at foreign STS in this case is obtained on base of presenting the
 983 SAML-Token of the Initiators STS in the according RST issue message. Depending on policies in
 984 effect, this SAML-Token may be replaced or cross-certified by applying a new signature. The SAML-
 985 Token key MUST be encrypted for the endpoint access is intended for. At the endpoint accessed,
 986 SAML-Token validation can be done on base of the signature of the foreign TD STS.

987 In the second trust model, the SAML-Token issued by the Initiator's STS directly is used for access
 988 authentication. In this case, the foreign endpoint points a RST validate message to his trusted STS for
 989 validating the foreign SAML-Token – what there again is done on base of SAML-Token signature,
 990 which must be trestud by the validating STS. Again, the SAML-Token key MUST be encrypted for the
 991 endpoint access is intended for.

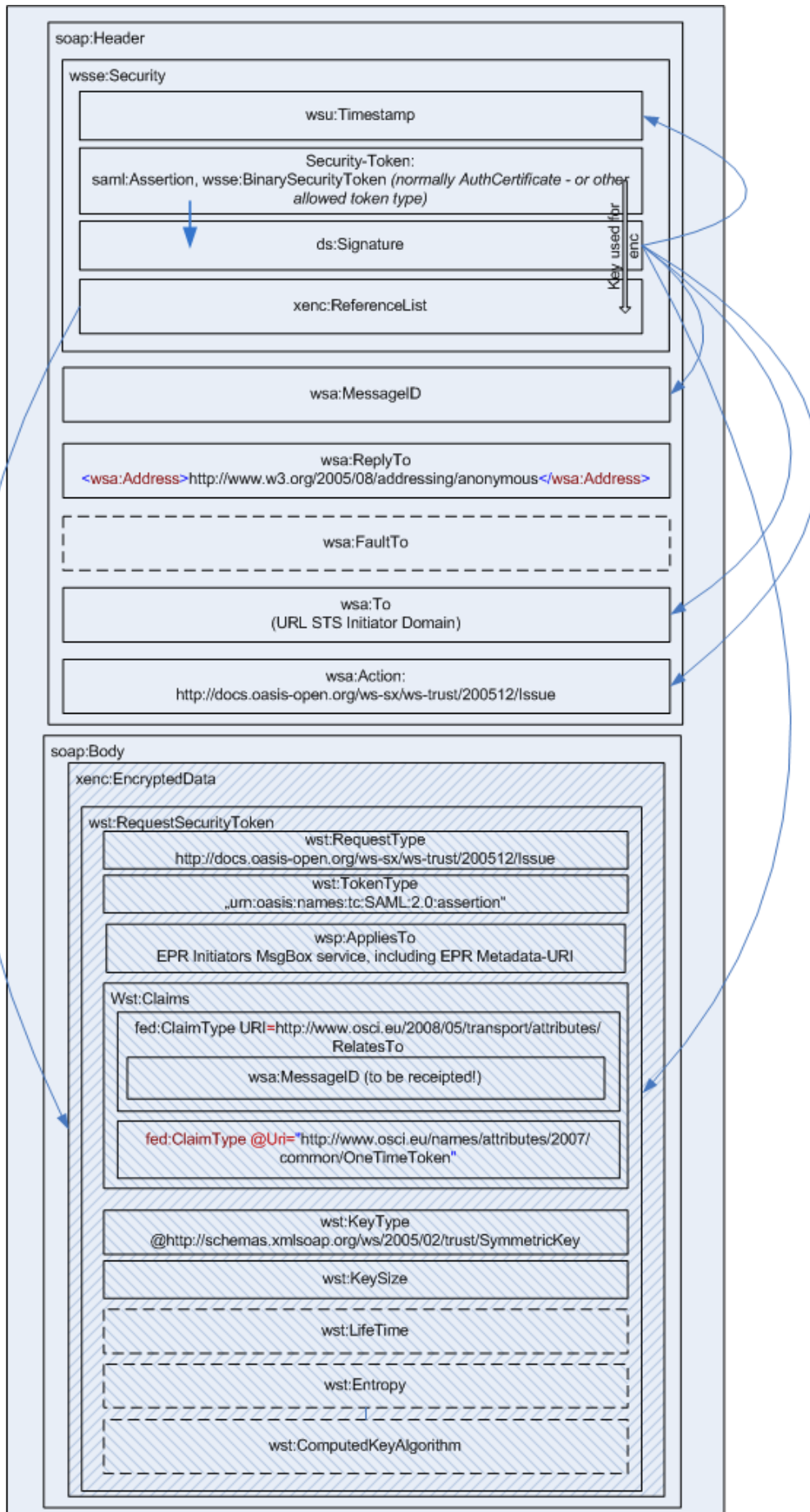
992 Details of the required SAML Token including claims and the issuing STS address as well as public
 993 key of this STS encryption certificate SHOULD be exposed by the endpoint WSDL. Apart, means of
 994 WS-Trust as already outlined in the chapters above apply.

995 **7.5.5 SAML-Token for Receipt- /Notification Delivery**

996 Requested receipts and notifications which cannot be delivered in the network backchannel of a
 997 request message MUST be delivered using an independent request message asynchronously to the
 998 EPR outlined in the receipt/notification request – which in general SHOULD be the Initiators MsgBox.
 999 As – like for all messages - delivery of receipts/notifications to this EPR requires authentication and
 1000 authorization, an according SAML-Token SHOULD be forwarded to the receipt/notification generating
 1001 node together with the request for it. This mechanism disburdens these nodes from the acquisition of
 1002 an extra SAML-Token to authenticate receipt/notification delivery.

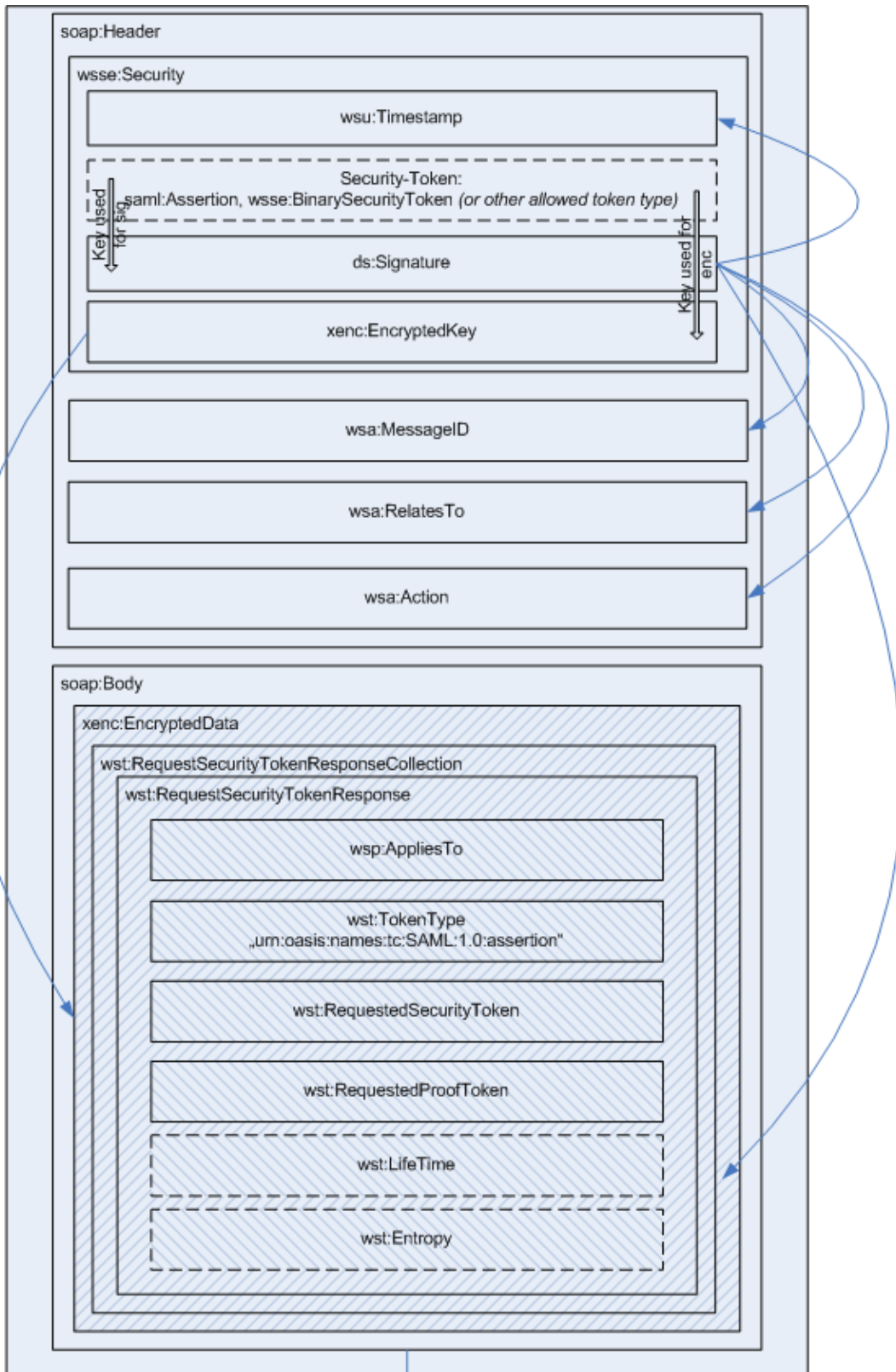
1003 This type of SAML-Token - referred to as "**OneTimeToken**" - is valid only for "one time use" of
 1004 receipt/notification delivery and bound to the `wsa:MessageID` of the message to be
 1005 receipted/notified. Following rules apply:

- 1006 1. It MUST be requested from the Initiators's STS.
- 1007 2. The according RST message MUST contain the **wsa:MessageID** and the address of the
1008 receiving/notifying node (**wsp:AppliesTo**) as claims.
- 1009 3. The symmetric key of the issued SAML-Token MUST be encrypted for the endpoint outlined in
1010 the element **.../wsa:ReplyTo** of the receipt/notification demand; the
1011 **wst:RequestedProofToken** in this case MUST be encrypted for receiving/notifying node
1012 (for use in step 6. ahead)
- 1013 4. The issuing STS MUST retain this OneTimeToken for later use and mark it as "unused".
- 1014 5. The RSTR message returned by the STS MUST be included as separate SOAP header block
1015 in the request message.
- 1016 6. The receiving/notifying node has to use the OneTimeToken included in this RSTR as SAML-
1017 Token for the message the receipt/notification is delivered with. Transport signature and
1018 encryption MUST be generated on base of the symmetric key contained in the
1019 **wst:RequestedProofToken**.
- 1020 Steps to be done by the node this message is targeted to:
- 1021 7. Decryption of the OneTimeToken's symmetric key.
- 1022 8. Validation of the signature of the OneTimeToken – the symmetric key MUST be the same the
1023 receiving/notifying node used for the transport signature.
- 1024 9. Validation of the signature of the issuing STS and RST-Validate message containing the
1025 OneTimeToken to the STS.
- 1026 10. If at the issuing STS this OneTimeToken is still marked as "unsed", the token is is valid.
- 1027 11. If RSTR in validate-request signals valid: Acceptance of the message containing the
1028 receipt/notification.
- 1029 12. Message accpeting node MUST generate a RST/cancel message to the STS to invalidate this
1030 OneTimeToken; STS SHOULD discard this token.
- 1031 Following diagrams illustrate the RST and RSTR for the OneTimeToken, for concrete XML Infoset
1032 descriptions see WS-Trust and SAML specifications.



1033

1034 Figure 6: RST for OneTimeToken



1035

1036 Figure 7: RSTR for OneTimeToken

1037 8 OSCI specific Extensions

1038 8.1 Message Flow Time Stamping

1039 For sake of traceability of message flow time instants and delivery status, every message of type
 1040 osci:Request MAY contain following SOAP header block, which child elements are provided
 1041 depending on the nodes passed in the message flow.

```
1042 <osci:MsgTimeStamps wsu:Id="..." ? >
1043   <osci:ObsoleteAfter> xs:date </osci:ObsoleteAfter> ?
1044   <osci:Delivery> xs:dateTime </osci:Delivery> ?
1045   <osci:InitialFetched> xs:dateTime </osci:InitialFetch> ?
1046   <osci:Reception> xs:dateTime </osci:Reception> ?
1047 </osci:MsgTimeStamps>
```

1048 Description of elements and attributes in the schema overview above:

1049 **/osci:MsgTimeStamps**

1050 This complex element is the container for various optional timestamp elements. It MUST
 1051 be created from the first node on the message flow which applies one or more sub-
 1052 elements.

1053 **/osci:MsgTimeStamps/@wsu:Id**

1054 For ease of referencing this SOAP header block from WS Security SOAP header
 1055 elements, this attribute of type `wsu:Id` SHOULD be provided.

1056 **/osci:MsgTimeStamps/osci:ObsoleteAfter ?**

1057 This element of type `xs:date` MAY be provided by an Initiator to denote the date after
 1058 which a message is to be seen as obsolete for delivery and/or consumption. It MUST NOT
 1059 be provided or changed by other nodes on the message path.

1060 If and how this information is handled by this endpoint this message is targeted to if
 1061 outlined in the policy of this endpoint; see chapter [10.2.2] for details.

1062 **/osci:MsgTimeStamps/osci:Delivery ?**

1063 This element of type `xs:dateTime` MUST be provided by a MsgBox node when
 1064 accepting an incoming message and MUST to be set to the value of the actual MsgBox
 1065 server time.

1066 It MUST NOT be provided or changed by other nodes on the message path.

1067 **/osci:MsgTimeStamps/osci:InitialFetched ?**

1068 This element of type `xs:dateTime` MUST be provided by a MsgBox node with to the
 1069 value of the actual MsgBox server time when an authorized Recipient initially pulls the
 1070 message from his MsgBox instance. commits the successful initial reception of this
 1071 message. This SHOULD be done by a Recipient after the first successful pulling of the
 1072 message from his MsgBox.

1073 It MUST NOT be provided or changed by other nodes on the message path. Pull
 1074 processes on the same message following a first confirmed one, this element MUST NOT
 1075 be updated.

1076 **/osci:MsgTimeStamps/osci:Reception ?**

1077 This element of type `xs:dateTime` MAY be set by a Recipient to his actual server time
 1078 when successfully accepting an incoming message, but it should be considered that the

1079 signature is invalidated which was applied over SOAP header and body elements by the
1080 message issuing instance.

1081 It MUST be set by a MsgBox node to his actual server time when the recipient commits
1082 the reception of a message thri a MsgBoxGetNextRequest or MsgBoxCloceRequest.

1083 It MUST NOT be provided or changed on the message path by other nodes than
1084 described here.

1085 8.2 Accessing message boxes

1086 Following chapters define how to access MsgBox services for searching and pulling out messages as
1087 well as how to gain status lists describing content of message boxes. Statuses of those requests are
1088 delivered in the SOAP header block of the correlating responses, while the pulled messages
1089 respective status lists are delivered in the SOAP body block.

1090 We describe the requests first, followed by the respective responses and additional messages to
1091 model "get next", "commit" and "close" semantics for iterative MsgBox access sequences.

1092 8.2.1 MsgBoxFetchRequest

1093 To request a message from an endpoint, a requestor MUST send a MsgBoxFetchRequest message to
1094 his MsgBox instance endpoint.

1095 The normative outline for a MsgBoxFetchRequest request is:

```

1096 <s12:Envelope ...>
1097   <s12:Header ...>
1098     ...
1099     <wsa:Action>
1100     http://www.osci.eu/ws/2008/05/transport/urn/messageTypes/MsgBoxFetchRequest
1101     </wsa:Action>
1102     <wsa:MessageID>xs:anyURI</wsa:MessageID>
1103     <wsa:ReplyTo>wsa:EndpointReference</wsa:ReplyTo>
1104     <wsa:To>xs:anyURI</wsa:To>
1105     <osci:TypeOfBusinessScenario @wsa:IsReferenceParameter="1">
1106     xs:anyURI
1107     </osci:TypeOfBusinessScenario>
1108     ...
1109   </s12:Header>
1110   <s12:Body ...>
1111     <osci:MsgBoxFetchRequest>
1112       <osci:EPR wsa:EndpointReference </osci:EPR>
1113       <osci:MsgSelector newEntry=("true" | "false")>
1114         <osci:MessageId> xs:anyURI </osci:MessageId> *
1115         <osci:RelatesTo> xs:anyURI </osci:RelatesTo> *
1116         <osci:MsgBoxEntryTimeFrom>
1117         xs:dateTime
1118         </osci:MsgBoxEntryTimeFrom> ?
1119         <osci:MsgBoxEntryTimeTo> xs:dateTime </osci:MsgBoxEntryTimeTo> ?
1120         <osci:Extension> xs:anyType </osci:Extension> ?
1121       </osci:MsgSelector> ?
1122     </osci:MsgBoxFetchRequest>
1123   </s12:Body>
1124 </s12:Envelope>

```

1125 The following describes normative constraints on the outline listed above:

1126 **/s12:Envelope/s12:Header/wsa:Action**

1127 The value indicated herein MUST be used for that URI.

1128 **/s12:Envelope/s12:Header/wsa:MessageID**

- 1129 The request MUST carry a unique WS-Addressing MessageID.
- 1130 `/s12:Envelope/s12:Header/wsa:ReplyTo`
- 1131 The Endpoint Reference (EPR) of the requesting Recipient endpoint. As
1132 MsgBoxFetchRequests are considered to only being meaningful in synchronous request-
1133 response MEPs, the EPRs address element MUST be restricted to the anonymous URI.
- 1134 `/s12:Envelope/s12:Header/wsa:To`
- 1135 The address of the MsgBox (request destination) endpoint.
- 1136 `/s12:Envelope/s12:Header/osci:TypeOfBusinessScenario`
- 1137 This value of the instantiation of `/wsa:ReferenceParameters` MUST be supplied for
1138 this message type. The value of `/osci:TypeOfBusinessScenario` is taken as
1139 message selection argument and SHOULD match one of those accepted by this endpoint.
- 1140 `/s12:Envelope/s12:Header/osci:TypeOfBusinessScenario/`
1141 `@wsa:IsReferenceParameter`
- 1142 Following WS-Addressing, the element MUST be attributed with
1143 `@wsa:IsReferenceParameter="1"`
- 1144 The body of this message contains the actual request in a structure
- 1145 `/s12:Envelope/s12:Body/osci:MsgBoxFetchRequest.`
- 1146 `/s12:Envelope/s12:Body/osci:MsgBoxFetchRequest/osci:EPR`
- 1147 The Endpoint Reference (EPR) of the message box endpoint where the requested
1148 messages are to be pulled from; type is `wsa:EndpointReferenceType`. This is the only
1149 element which MUST be present in a `MsgBoxFetchRequest`. Only messages originally
1150 targeted to this EPR MUST be returned. The EPR MUST contain reference properties to
1151 help uniquely identify the source endpoint according to chapter [6, Addressing Endpoints].
- 1152 If a `MsgBoxFetchRequest` contains no other arguments for message selection, the
1153 messages to be selected MUST be those which have not yet been fetched. Those are all
1154 messages in the addressed `MsgBox` which have no SOAP header element or a value of
1155 zero in the time instant element `.../osci:MsgTimeStamps/osci:InitialFetches`.
1156 They MUST be delivered one per request-/ response in a FIFO-manner to the endpoint
1157 denoted by `/s12:Envelope/s12:Header/wsa:ReplyTo`.
- 1158 `/s12:Envelope/s12:Body/osci:MsgBoxFetchRequest/osci:MsgSelector ?`
- 1159 If this optional element is present, arguments of the attribute `@newEntry` and sub-
1160 elements `MsgSelector` MUST be processed as logical AND (after first OR-Processing of
1161 the sequences of MessageIDs in the SOAP body elements `.../osci:MessageID` and
1162 `.../osci:RelatesTo`, if present).
- 1163 `/s12:Envelope/s12:Body/osci:MsgBoxFetchRequest/osci:MsgSelector/@newEntry ?`
- 1164 This optional boolean attribute is defaulted to a value of true, if not present. If present, this
1165 attribute denotes whether only already pulled or new entered messages have to be
1166 selected from the `MsgBox`. "New" messages are indicated by having no SOAP header
1167 element or a value of zero in the time instant element
1168 `.../osci:MsgTimeStamps/osci:InitialFetches`.
- 1169 `/s12:Envelope/s12:Body/osci:MsgBoxFetchRequest/osci:MsgSelector/`
1170 `osci:MessageID *`
- 1171 If present, this element contains a sequence of WS-Addressing MessageIDs. By including
1172 this element the request a `MsgBox` service MUST limit its search to just those messages
1173 with these values in the WS-Addressing SOAP header element `.../wsa:MessageID`.

- 1174 `/s12:Envelope/s12:Body/osci:MsgBoxFetchRequest/osci:MsgSelector/`
 1175 `osci:RelatesTo *`
- 1176 If present, this element contains a sequence of WS-Addressing MessageIDs. By including
 1177 this element the request a MsgBox service MUST limit its search to just those messages
 1178 with these values in the WS-Addressing SOAP header elements `.../wsa:RelatesTo`.
- 1179 `/s12:Envelope/s12:Body/osci:MsgBoxFetchRequest/osci:MsgSelector/`
 1180 `osci:MsgBoxEntryTimeFrom ?`
- 1181 If present, this element denotes a value of type `xs:dateTime` as lower value when a
 1182 message has been accepted by a MsgBox service. The resulting search expression is
 1183 `.../osci:MsgBoxEntryTimeFrom >=` the value of `.../osci:Delivery` in the message
 1184 SOAP header block `osci:MsgTimeStamps` if the correlated element
 1185 `.../osci:MsgBoxEntryTimeTo` is not present in `.../osci:MsgSelector`.
- 1186 `/s12:Envelope/s12:Body/osci:MsgBoxFetchRequest/osci:MsgSelector/`
 1187 `osci:MsgBoxEntryTo ?`
- 1188 If present, this element denotes a value of `xs:dateTime` as upper value when a
 1189 message has been accepted by a MsgBox services. The resulting search expression is
 1190 `.../osci:MsgBoxEntryTimeTo <=` the value of `.../osci:Delivery` in the message
 1191 SOAP header block `osci:MsgTimeStamps` if the correlated element
 1192 `.../MsgBoxEntryTimeFrom` is not present in `.../osci:MsgSelector`.
- 1193 If latter elements both are set, the resulting search expression is
 1194 `.../osci:MsgBoxEntryFrom >= .../osci:Delivery <=`
 1195 `.../osci:MsgBoxEntryTimeTo`; the value of `.../osci:MsgBoxEntryFrom` MUST be
 1196 less or equal to the value of `.../osci:MsgBoxEntryTo`.
- 1197 `/s12:Envelope/s12:Body/osci:MsgBoxFetchRequest/osci:MsgSelector/`
 1198 `osci:Extension/{any} *`
- 1199 This is an extensibility mechanism to allow other search criteria to be passed. For
 1200 example, an XPath query could be used to search for messages that match a certain
 1201 pattern. Implementations may use this element for defining search criteria on agreements
 1202 outbound to this specification. At some future time this specification will define such an
 1203 extension.
- 1204 Upon receipt and authentication of this message, the MsgBox service MUST locate any message that
 1205 matches the selection criteria. Only messages originally targeted to this EPR MUST be returned. The
 1206 search criteria MUST include examination of the child elements inside the SOAP body element
 1207 `.../osci:MsgSelector`.
- 1208 Selected messages MUST be given back to the requestor one by one in the response to this request
 1209 in an ascending order given by the values of the SOAP header block element
 1210 `/osci:MsgTimeStamps/osci:Delivery` ("FIFO"). A MsgBox service MUST hold the complete
 1211 list corresponding to the selection criteria and deliver an ID for this list to the requestor with the
 1212 response. In subsequent requests (see `MsgBoxGetNextRequest` in chapter [8.2.4]) the Requestor is
 1213 able to pull further messages of a selection result with reference to this list. Remaining messages of
 1214 the complete list MUST be retained for following messages of type `MsgBoxGetNextRequest`.
- 1215 **8.2.2 MsgBoxStatusListRequest**
- 1216 To request a message status list from a MsgBox service endpoint, a requestor MUST send a
 1217 `MsgBoxStatusListRequest` message to his MsgBox instance endpoint.
- 1218 The normative outline for a `MsgBoxStatusListRequest` request is akin to the `MsgBoxFetchRequest`:
- 1219 `<s12:Envelope ...>`
 1220 `<s12:Header ...>`
 1221 `...`

```

1222     <wsa:Action>
1223 http://www.osci.eu/ws/2008/05/transport/urn/messageTypes/MsgBoxStatusListRe
1224 quest
1225     </wsa:Action>
1226     <wsa:MessageID>xs:anyURI</wsa:MessageID>
1227     <wsa:ReplyTo> wsa:EndpointReference </wsa:ReplyTo>
1228     <wsa:To>xs:anyURI</wsa:To>
1229     <osci:TypeOfBusinessScenario @wsa:IsReferenceParameter="1">
1230       xs:anyURI
1231     </osci:TypeOfBusinessScenario>
1232
1233     ...
1234 </s12:Header>
1235 <s12:Body ...>
1236   <osci:MsgBoxStatusListRequest maxListItems="xs:positiveInteger">
1237     <osci:EPR> wsa:EndpointReference </osci:EPR>
1238     <osci:MsgSelector newEntry=("true" | "false")>
1239       <osci:MessageId> xs:anyURI </osci:MessageId> *
1240       <osci:RelatesTo> xs:anyURI </osci:RelatesTo> *
1241       <osci:MsgBoxEntryTimeFrom>
1242         xs:dateTime
1243       </osci:MsgBoxEntryTimeFrom> ?
1244       <osci:MsgBoxEntryTimeTo>
1245         xs:dateTime
1246       </osci:MsgBoxEntryTimeTo> ?
1247       <osci:Extension> xs:anyType </osci:Extension> ?
1248     </osci:MsgSelector> ?
1249   </osci:MsgBoxStatusListRequest>
1250 </s12:Body>
1251 </s12:Envelope>

```

1252 Description of normative constraints on the outline listed above:

1253 **/s12:Envelope/s12:Header/wsa:Action**

1254 The value indicated herein MUST be used for that URI.

1255 **/s12:Envelope/s12:Header/wsa:MessageID**

1256 The request MUST carry a unique WS-Addressing MessageID.

1257 **/s12:Envelope/s12:Header/wsa:ReplyTo**

1258 The Endpoint Reference (EPR) of the requesting Recipient endpoint. As message box
1259 requests considered to only being meaningful in synchronous request-response message
1260 MEPs, the EPRs address element MUST be restricted to the anonymous URI.

1261 **/s12:Envelope/s12:Header/wsa:To**

1262 The address of the MsgBox (request destination) endpoint.

1263 **/s12:Envelope/s12:Header/osci:TypeOfBusinessScenario**

1264 This value of the instantiation of **/wsa:ReferenceParameters** MUST be supplied for
1265 this message type. The value of **/osci:TypeOfBusinessScenario** is taken as
1266 message selection argument and SHOULD match one of those accepted by this endpoint.
1267 To select the message status list for all messages in this MsgBox instance, a value of ""
1268 MAY be supplied here.

1269 **/s12:Envelope/s12:Header/osci:TypeOfBusinessScenario/
1270 @wsa:IsReferenceParameter**

1271 Following WS-Addressing, the element MUST be attributed with
1272 **@wsa:IsReferenceParameter="1"**

1273 The body of this message contains the actual request in the structure

1274 **/s12:Envelope/s12:Body/osci:MsgBoxStatusListRequest.**

- 1275 `/s12:Envelope/s12:Body/osci:MsgBoxStatusListRequest/@maxListItems ?`
- 1276 The requestor MAY limit the length of the message status list he expects in the response
 1277 with this attribute of type `xs:positiveInteger`. A MsgBox service MUST hold the
 1278 complete list corresponding to the selection criteria and deliver an ID for this list to the
 1279 requestor with the response. In subsequent requests (see `MsgBoxGetNextRequest` in
 1280 chapter [8.2.4]), the requestor is able to request further portions of a selection result with
 1281 reference to this list.
- 1282 A MsgBox instance MAY limit the value of `@maxListItems` to any value greater zero.
- 1283 If provided, a MsgBox instance MUST retain this value – if not decreased by its configured
 1284 limit - together with the result set until the whole result set is delivered to the requestor or
 1285 the requestor cancels an iteration sequence (see `MsgBoxCloseRequest` in chapter [8.2.5].
- 1286 `/s12:Envelope/s12:Body/osci:MsgBoxStatusListRequest/osci:EPR`
- 1287 The EPR of the message box endpoint where the requested message status list has to be
 1288 pulled from; type is `wsa:EndpointReferenceType`. This is the only element which
 1289 MUST be present in a `MsgBoxFetchRequest`. Only status lists of messages originally
 1290 targeted to this EPR MUST be returned. The EPR MUST contain
 1291 `wsa:ReferenceParameters` to help uniquely identify the source endpoint according to
 1292 chapter [6, Addressing Endpoints]. If a `MsgBoxFetchRequest` contains no other
 1293 arguments for message selection, the messages to be selected MUST be those which
 1294 have not yet been fetched. That are all messages in the addressed MsgBox having no
 1295 SOAP header element or a value of zero in the
 1296 `.../osci:MsgTimeStamps/osci:InitialFetched` time instant element.
- 1297 While for a `.../osci:MsgBoxFetchRequest` the EPR has to be specified completely
 1298 including the type of business scenario in `.../wsa:ReferenceParameters` according to
 1299 chapter [6, Addressing Endpoints], for the `MsgBoxStatusListRequest` it is possible to
 1300 supply the element
 1301 `.../wsa:ReferenceParameters/osci:TypeOfBusinessScenario` with a value of
 1302 `"*"`. This entry MUST lead to a message box status list containing all messages with no
 1303 regard to a specific addressed business scenario of this endpoint actually exposes to be
 1304 able to serve.
- 1305 `/s12:Envelope/s12:Body/osci:MsgBoxStatusListRequest/osci:MsgSelector`
- 1306 For the content of this complex element, which defines selection criteria for messages,
 1307 see description in last chapter [8.2.1].
- 1308 `/s12:Envelope/s12:Body/osci:MsgBoxStatusListRequest/osci:MsgSelector/
 1309 osci:Extension/{any} *`
- 1310 This is an extensibility mechanism to allow other search criteria to be passed. See
 1311 respective explanation for `osci:MsgBoxStatusListRequest`.
- 1312 Upon receipt and authentication of this message, the MsgBox service will locate any message that
 1313 matches the selection criteria. Only messages originally targeted to this EPR MUST be selected for
 1314 the required message status list. The search criteria MUST include examination of the child elements
 1315 inside the `/osci:MsgSelector` SOAP body element.
- 1316 The message status list to be given back to the requestor MUST be of the maximum size denoted by
 1317 `/osci:MsgBoxStatusListRequest/@maxListItems` or a lower size according to possible
 1318 configured restrictions of the requested MsgBox instance. The list MUST be build up and sorted in an
 1319 ascending order given by the message SOAP header block element
 1320 `/osci:MsgTimeStamps/osci:Delivery` ("FIFO"). Remaining items of the complete list not
 1321 deliverable to the requestor directly in the response to the initial `MsgBoxStatusListRequest` MUST be
 1322 retained for following messages of type `MsgBoxGetNextRequest`.

1323 8.2.3 MsgBoxResponse

1324 Request messages MsgBoxFetchRequest und MsgBoxStatusListRequest are both responded by the
1325 same status information in the SOAP header block of the response, only the body parts differ as
1326 outlined in the following chapters.

1327 The normative outline for a MsgBoxResponse header is:

```

1328 <s12:Envelope ...>
1329   <s12:Header ...>
1330     ...
1331     <wsa:Action>
1332       http://www.osci.eu/ws/2008/05/transport/urn/messageTypes/MsgBoxResponse
1333     </wsa:Action>
1334     <wsa:MessageID>xs:anyURI</wsa:MessageID>
1335     <wsa:FaultTo> wsa:EndpointReference </wsa:FaultTo> ?
1336     <wsa:To>xs:anyURI</wsa:To>
1337     ...
1338     <osci:MsgBoxResponse MsgBoxRequestID="xs:anyURI"
1339       wsu:Id = "xs:ID" ?
1340       <osci:NoMessageAvailable reason=xs:QName/>
1341       |
1342       <osci:ItemsPending resultIteratorID=xs:positiveInteger ?
1343         xs:positveInteger
1344       </osci:ItemPending>
1345     </osci:MsgBoxResponse>
1346     ...
1347   </s12:Header>
1348   <s12:Body ...>
1349
1350 </s12:Body>
1351 </s12:Envelope>

```

1352 Description of normative constraints on the outline listed above (WS-Addressing header elements are
1353 to be handled according to chapter [6, Addressing Endpoints]):

1354 **/s12:Envelope/s12:Header/wsa:Action**

1355 The value indicated herein MUST be used for that URI.

1356 **/s12:Envelope/s12:Header/wsa:MessageID**

1357 The request MUST carry a unique WS-Addressing MessageID.

1358 **/s12:Envelope/s12:Header/wsa:FaultTo ?**

1359 The optional Endpoint Reference (EPR) SOAP fault messages should be routed to.

1360 **/s12:Envelope/s12:Header/wsa:To**

1361 The address of the endpoint of the initial requestor, derived from the **/wsa:ReplyTo**
1362 header element of the corresponding MsgBox request.

1363 **/s12:Envelope/s12:Header/osci:MsgBoxResponse**

1364 The container for the status response; the status response is a choice of two alternatives,
1365 depending on request fulfilment.

1366 Following attributes MUST be set:

1367 **/s12:Envelope/s12:Header/osci:MsgBoxResponse/@MsgBoxRequestID**

1368 This mandatory element of type **xs:anyURI** MUST carry a unique value of type UUID
1369 according to [RFC4122]. It serves to identify messages of type MsgBoxFetchRequest and
1370 MsgBoxStatusListRequest and MUST be used by the Requestor in subsequent messages
1371 of type MsgBoxGetNextRequest or MsgBoxCloseRequest explained below. The value
1372 MUST be generated by a MsgBox instance for every incoming MsgBoxFestRequest or

1373 MsgBoxStatusListRequest and MUST be retained and correlated to these requests with
1374 their individual search criteria.

1375 /*s12:Envelope/s12:Header/osci:MsgBoxResponse/@wsu:Id* ?

1376 For ease of referencing this SOAP body block, this optional attribute of type *wsu:Id* MAY
1377 be provided.

1378 /*s12:Envelope/s12:Header/osci:MsgBoxResponse/osci:NoMessageAvailable*

1379 This element of the choice of . . . /*MsgBoxResponse* MUST be set if

- 1380 - there are no messages available corresponding to the selection criteria
- 1381 - there where errors detected in the selection criteria.

1382 The element carries following attribute:

1383 /*s12:Envelope/s12:Header/osci:MsgBoxResponse/osci:NoMessageAvailable/*
1384 *@reason* ?

1385 Attribute of type *xs:QName*, identifies the reason of /*osci:NoMessageAvailable* set.

1386 The alternate of the choice of . . . /*MsgBoxResponse* MUST be set if there are – according to the
1387 selection criteria of the request - messages or message status list items pending (not yet deliverable
1388 to the requestor in the actual response):

1389 /*s12:Envelope/s12:Header/osci:MsgBoxResponse/osci:ItemsPending*

1390 This element of type *xs:nonNegativeInteger* MUST be set with the number of the
1391 remaining items. If the last portion of a result set delivered to the requestor with this actual
1392 response, the value MUST be set to zero to signal this fact.

1393 **8.2.3.1 MsgBoxResponse - body to MsgBoxFetchRequest**

1394 For this type of foregoing initial request, the requested message MUST be delivered in following
1395 manner:

- 1396 • A SOAP Envelope with all child elements MUST be build up containing a header block with
1397 the ones of the original message except header elements which had initially been targeted to
1398 and successfully executed by the MsgBox node as well as the transport encryption and
1399 signature elements which had been supplied by the Initiator of this message.
- 1400 • The SOAP header element *osci:MsgTimeStamps* MUST be inserted (or completed, if
1401 present) as described in chapter [8.1].
- 1402 • All original WS-Addressing, *osci:X509TokenContainer* , *xkms:ValidateResult* (inside
1403 a *xkms:CompoundResult*) and original Security Token and WS-Trust header elements
1404 MUST be included.
- 1405 • If present in the original message, the *osci:ReceptionReceiptDemand* header element
1406 MUST be included.
- 1407 • The original SOAP body child elements MUST be included unchanged as child elements of
1408 the SOAP body of this SOAP Envelope to be build up.
- 1409 • The Recipient may have interest in the authentication and authorization data originally carried
1410 in the SOAP WS Security header when delivering a message to the Recipients MsgBox.
1411 Therefore, this security token MUST be inserted as additional child element in the SOAP
1412 header of this SOAP Envelope to be built up.

1413 The resulting SOAP envelope MUST be included as child element of the SOAP body block of the
1414 response message.

1415 **8.2.3.2 MsgBoxResponse - body to MsgBoxStatusListRequest**

1416 For this type of foregoing initial request, the requested list MUST be build up in the SOAP body block
 1417 of the response message. This is the same for responses to subsequent requests of type
 1418 MsgBoxGetNextRequest (see MsgBoxGetNextRequest in chapter [8.2.4]).

1419 The normative outline for a MsgStatusList is:

```

1420 <osci:MsgStatusList>
1421   <osci:MsgAttributes>
1422     <wsa:MessageID>xs:anyUri</wsa:MessageID>
1423     <wsa:RelatesTo>xs:anyUri</wsa:RelatesTo> *
1424     <wsa:From>endpoint-reference</wsa:From>
1425     <osci:TypeOfBusinessScenario>xs:anyUri</osci:TypeOfBusinessScenario>
1426     <wsa:Action>xs:anyUri</wsa:Action>
1427     <osci:MsgSize>xs:positiveInteger</osci:msgSize>
1428     <osci:ObsoleteAfterDate> xs:date </osci:ObsoleteAfterDate> ?
1429     <osci:DeliveryTime> xs:dateTime </osci:DeliveryTime>
1430     <osci:InitialFetchTime> xs:dateTime </osci:InitialFetchTime> ?
1431   </osci:MsgAttributes> *
1432 </osci:MsgStatusList>
  
```

1433 The whole structure MUST be positioned under `/s12:Envelope/s12:Body`.

1434 `/osci:MsgStatusList`

1435 Container for the items of the list.

1436 `/osci:MsgStatusList/osci:MsgAttributes *`

1437 The container for the attributes of one message of the status list. The number of
 1438 occurrences is determined by the number of items of selection result list not yet delivered
 1439 to the requestor and the value of

1440 `.../osci:MsgBoxStatusListRequest/@maxListItems` of the initial
 1441 MsgBoxStatusListRequest, which MAY be modified to a lower value greater zero set by
 1442 the requested MsgBox instance.

1443 `/osci:MsgStatusList/osci:MsgAttributes/wsa:MessageID`

1444 MessageID of the message, derived from respective header element.

1445 `/osci:MsgStatusList/osci:MsgAttributes/wsa:RelatesTo *`

1446 MessageIDs of related messages of the message, derived from respective header
 1447 elements.

1448 `/osci:MsgStatusList/osci:MsgAttributes/wsa:From`

1449 From-EPR of the message, derived from the respective header element.

1450 `/osci:MsgStatusList/osci:TypeOfBusinessScenario`

1451 This URI denotes the type of addressed business scenario of the intended recipient of the
 1452 message. It is derived from the `/wsa:ReferenceParameters`
 1453 `/osci:TypeOfBusinessScenario` associated to the WS-Addressing SOAP header
 1454 element `/wsa:To` of the message.

1455 `/osci:MsgStatusList/wsa:Action`

1456 Derived from respective SOAP header element `/wsa:Action` set in the message.

1457 `/osci:MsgStatusList/osci:MsgSize`

1458 The size of the message in kilobytes has to be supplied here as `xs:positiveInteger`.

1459 Following timestamps are provided in an `osci:MsgStatusList` according to the

1460 `/osci:MsgTimeStamps` described in chapter [8.1]:

1461 `/osci:MsgStatusList/ObsoleteAfterDate ?`

- 1462 Optional element of type `xs:date`, contains - if present in the underlying message - the
 1463 value of the SOAP header block element
 1464 `/osci:MsgTimeStamps/osci:ObsoleteAfter` present in the underlying message.
- 1465 `/osci:MsgStatusList/DeliveryTime`
- 1466 This element of type `xs:dateTime` contains the value of the SOAP header block element
 1467 `.../osci:MsgTimeStamps/osci:Delivery`, which MUST be present in a message
 1468 stored by a `MsgBox` instance.
- 1469 `/osci:MsgStatusList/InitialFetchedTime ?`
- 1470 This optional element of type `xs:dateTime` contains the value of the SOAP header block
 1471 element `.../osci:MsgTimeStamps/osci:InitialFetched`, which MAY be present in
 1472 a message stored by a `MsgBox` instance. Only if not present or present with a value of
 1473 zero, the `MsgBox` instance MUST provide this element with his actual server time before
 1474 message delivery.

1475 8.2.4 MsgBoxGetNextRequest

- 1476 To request subsequent, not yet delivered results from foregoing requests of type
 1477 `MsgBoxStatusListRequest` or `MsgBoxFetchRequest`, a requestor MUST send a
 1478 `MsgBoxGetNextRequest` message to the same `MsgBox` instance.

- 1479 The normative outline for a `MsgBoxGetNextRequest`:

```

1480 <s12:Envelope ...>
1481   <s12:Header ...>
1482     ...
1483     <wsa:Action>
1484       http://www.osci.eu/ws/2008/05/transport/urn/messageTypes/
1485       MsgBoxGetNextRequest
1486     </wsa:Action>
1487     <wsa:MessageID>xs:anyURI</wsa:MessageID>
1488     <wsa:ReplyTo>endpoint-reference</wsa:ReplyTo>
1489     <wsa:To>xs:anyURI</wsa:To>
1490     <osci:TypeOfBusinessScenario @wsa:IsReferenceParameter="1">
1491       xs:anyURI
1492     </osci:TypeOfBusinessScenario>
1493   ...
1494   </s12:Header>
1495   <s12:Body ...>
1496     <osci:MsgBoxGetNextRequest MsgBoxRequestID="xs:anyURI">
1497       <osci:EPR>MsgBox EPR</osci:EPR>
1498       <osci:LastMsgReceived> wsa:MessageID </osci:LastMsgReceived> *
1499     </osci:MsgBoxGetNextRequest>
1500   </s12:Body>
1501 </s12:Envelope>
  
```

- 1503 Description of normative constraints on the outline listed above:

1504 `/s12:Envelope/s12:Header/wsa:Action`

- 1505 The value indicated herein MUST be used for that URI.

1506 `/s12:Envelope/s12:Header/wsa:MessageID`

- 1507 The request MUST carry a unique WS-Addressing MessageID.

1508 `/s12:Envelope/s12:Header/wsa:ReplyTo`

- 1509 The EPR of the requesting (source-) endpoint. As message box requests considered to
 1510 only being meaningful in synchronous request-response scenarios, EPRs address
 1511 element MUST be restricted to the anonymous URI.

1512 `/s12:Envelope/s12:Header/wsa:To`

- 1513 The address of the `MsgBox` (request destination) endpoint.

- 1514 `/s12:Envelope/s12:Header/osci:TypeOfBusinessScenario`
- 1515 The corresponding value of the initial `MsgBoxFetchRequest` or `MsgBoxStatusListRequest`
- 1516 MUST be supplied for this message type.
- 1517 `/s12:Envelope/s12:Header/osci:TypeOfBusinessScenario/`
- 1518 `@wsa:IsReferenceParameter`
- 1519 Following WS-Addressing, the element MUST be attributed with
- 1520 `@wsa:IsReferenceParameter="1"`
- 1521 The body of this message contains the actual
- 1522 `/s12:Envelope/s12:Body/osci:MsgBoxGetNextRequest.`
- 1523 `/s12:Envelope/s12:Body/osci:MsgBoxGetNextRequest/@MsgBoxRequestID`
- 1524 This attribute of type `xs:anyURI` MUST be provided with the value of the with this
- 1525 message referenced foregoing `MsgBoxResponse/@MsgBoxRequestID`. The `MsgBox`
- 1526 service MUST use it to correlate this `MsgBoxGetNextRequest` to the initial
- 1527 `MsgBoxFetchRequest` respective `MsgBoxStatusListRequest`.
- 1528 `/s12:Envelope/s12:Body/osci:MsgBoxGetNextRequest/osci:EPR`
- 1529 The EPR of the message box endpoint where the next requested message status list
- 1530 respective message has to be pulled from, type is `wsa:EndpointReferenceType`.
- 1531 This element SHOULD be interpreted from the `MsgBox` service for verifying the same
- 1532 EPR is still requested as it has been in the initial request – if consistence of such request
- 1533 sequences is not secured by other mechanisms.
- 1534 `/s12:Envelope/s12:Body/osci:MsgBoxGetNextRequest/osci:LastMsgReceived *`
- 1535 These optional elements of type `wsa:AttributedURIType` MAY be provided, when the
- 1536 underlying initial request was of the type `MsgBoxFetchRequest`. The requestor SHOULD
- 1537 provide here the value(s) of the `/wsa:MessageID` of the last message(s) he received in
- 1538 the body of the foregoing response(s) to commit successful reception of those messages.
- 1539 This has to be realized as "reception acknowledge by requester" by the `MsgBox` instance:
- 1540 If the SOAP header element `.../osci:MsgTimeStamps/osci:Reception` is absent or
- 1541 the value of the SOAP header element `.../osci:MsgTimeStamps/osci:Reception` is
- 1542 zero, the actual server time of the `MsgBox` instance MUST now be set here and the value
- 1543 has to be signed according to chapter [8.1]. The resulting changes in the SOAP header
- 1544 block `/osci:MsgTimeStamps` now MUST be persisted in the `MsgBox` store.
- 1545 Upon receipt and authentication of this message, the `MsgBox` service MUST – depending on type of
- 1546 initial request referenced by `osci:MsgBoxGetNextRequest/@MsBoxRequestID`
- 1547 – deliver a `MsgBoxResponse` with the next message of the list indicated by
- 1548 `/osci:MsgBoxResponse/@MsBoxRequestID` in the body of the response (rules denoted
- 1549 in chapter [8.2.3.1] apply)
- 1550 – deliver a `MsgBoxResponse` with the next portion of a `/osci:MsgStatusList` indicated by
- 1551 `/osci:MsgBoxResponse/@MsBoxRequestID` in the body of the response (rules denoted
- 1552 in chapter [8.2.3.2] apply).
- 1553 Inside the SOAP header element `.../osci:MsgBoxResponse`, choice `.../osci:ItemsPending`
- 1554 MUST be set to the actual value. If `.../osci:ItemsPending` becomes a value auf zero now, this fact
- 1555 signal the requestor that the `MsgBox` instance may have discarded the search result list referenced by
- 1556 the identifier `/osci:MsgBoxResponse/@MsBoxRequestID`.

1557 **8.2.5 MsgBoxCloseRequest**

1558 The functionalities of this message type are:

- 1559 • Recipient commits successful reception of messages from his `MsgBox` instance

- 1560 • Recipient signals the abortion of an iterative pull process of sequences of result requests of
1561 foregoing initial `MsgBoxFetchRequest` respective `MsgBoxStatusListRequest`.

1562 NOTE: In case of successful processing of a `MsgBoxCloseRequest` by the targeted `MsgBox` instance a
1563 response MUST NOT be generated.

1564 The normative outline for the `MsgBoxCloseRequest`:

```
1565 <s12:Envelope ...>
1566   <s12:Header ...>
1567     ...
1568     <wsa:Action>
1569     http://www.osci.eu/ws/2008/05/transport/urn/messageTypes/MsgBoxCloseRequest
1570     </wsa:Action>
1571     <wsa:MessageID>xs:anyURI</wsa:MessageID>
1572     <wsa:ReplyTo>endpoint-reference</wsa:ReplyTo>
1573     <wsa:To>xs:anyURI</wsa:To>
1574     <osci:TypeOfBusinessScenario @wsa:IsReferenceParameter="1">
1575     xs:anyURI
1576     </osci:TypeOfBusinessScenario>
1577     ...
1578   </s12:Header>
1579   <s12:Body ...>
1580     <osci:MsgBoxCloseRequest MsgBoxRequestID="xs:anyURI">
1581       <osci:LastMsgReceived>xs:anyURI</LastMsgReceived> *
1582     </osci:MsgBoxCloseRequest>
1583   </s12:Body>
1584 </s12:Envelope>
```

1586 Description of normative constraints on the outline listed above:

1587 **/s12:Envelope/s12:Header/wsa:Action**

1588 The value indicated herein MUST be used for that URI.

1589 **/s12:Envelope/s12:Header/wsa:MessageID**

1590 The request MUST carry a unique WS-Addressing MessageID.

1591 **/s12:Envelope/s12:Header/wsa:ReplyTo**

1592 The EPR of the requesting (source-) endpoint. As message box requests considered to
1593 only being meaningful in synchronous request-response scenarios, EPRs Address
1594 element MUST be restricted to the anonymous URI.

1595 **/s12:Envelope/s12:Header/wsa:To**

1596 The address of the `MsgBox` (request destination) endpoint.

1597 **/s12:Envelope/s12:Header/osci:TypeOfBusinessScenario**

1598 The corresponding value of the initial `MsgBoxFetchRequest` or `MsgBoxStatusListRequest`
1599 MUST be supplied for this message type.

1600 **/s12:Envelope/s12:Header/osci:TypeOfBusinessScenario/**

1601 **@wsa:IsReferenceParameter**

1602 Following WS-Addressing, the element MUST be attributed with

1603 **@wsa:IsReferenceParameter="1"**

1604 The body of this message contains the actual

1605 **/s12:Envelope/s12:Body/osci:MsgBoxCloseRequest.**

1606 **/s12:Envelope/s12:Body/osci:MsgBoxGetNextRequest/@MsgBoxRequestID**

1607 This attribute of type `xs:anyURI` MUST be provided with the value of the with this
1608 message referenced foregoing `MsgBoxResponse/@MsgBoxRequestID`. The `MsgBox`

1609 service MUST use it to correlate this `MsgBoxCloseRequest` to the initial
1610 `MsgBoxFetchRequest` respective `MsgBoxStatusListRequest`.

1611 `/s12:Envelope/s12:Body/osci:MsgBoxCloseRequest/LastMsgReceived ?`

1612 These optional elements of type `wsa:AttributedURIType` MAY be provided, when the
1613 underlying initial request was of the type `MsgBoxFetchRequest`. The requestor SHOULD
1614 provide here the value(s) of the `/wsa:MessageID` of the last message(s) he received in
1615 the body of the foregoing response(s) to commit successful reception of those messages.
1616 This has to be realized as "reception acknowledge by requester" by the `MsgBox` instance:
1617 If the SOAP header element `.../osci:MsgTimeStamps/osci:InitialFetched` is
1618 absent or present with a value of zero, the actual server time of the `MsgBox` instance
1619 MUST now be set here and the value has to be signed according to chapter [8.1]. The
1620 resulting changes in the SOAP header block `/osci:MsgTimeStamps` now MUST be
1621 persisted in the `MsgBox` store.

1622 It should be noted, that this message type MUST be send to the `MsgBox` instance ever
1623 when a `MsgBoxFetchRequest` was the initial message send and the requestor pulled a
1624 message successfully the first time (a new message). This triggers the commitment of the
1625 SOAP header `/osci:MsgTimeStamps/osci:Reception` and
1626 `/osci:MsgTimeStamps/osci:InitialFetched` time instances. *It is up to*
1627 *implementations of recipient instances, how distinct between "new" and already*
1628 *processed messages*, because the recipient transport gateway has no implicit control on
1629 the state of the successful body processing (for example to mark message as "readed" or
1630 "processed" – this at least is under the control of the application targeted by the
1631 message).

1632 To avoid situations, where successful pulled messages on the `MsgBox` instance side
1633 remain in the state unpulled, it is strongly recommended to commit every
1634 `MsgBoxResponse` to an initial `MsgBoxFetchRequest` and following series of
1635 `MsgBoxGetNextRequest`.

1636 8.2.6 Processing Rules for `MsgBoxGetNext/CloseRequest`

1637 `MsgBox` instances are free to to configure a timeout value to retain search result list identified by
1638 `/osci:MsgBoxResponse/@MsgBoxRequestID`.

1639 If a `MsgBox` instance receives a `MsgBoxGetNextRequest` or a `MsgBoxCloseRequest` not at all or not
1640 more known here, no processing on the message database must be done and a following fault MUST
1641 be generated:

1642 Fault 7: **`MsgBoxRequestWrongReference`**

1643 [Code] Sender

1644 [Subcode] `MsgBoxRequestWrongReference`

1645 [Reason] `MsgBoxRequestID` unknown or timed out.

1646 8.3 Receipts

1647 Requirements for receipting message exchange were outlined in "OSCI-Transport 2.0 – Functional
1648 Requirements and Design Objectives" and "OSCI-Transport 2 – Technical Features Overview"

1649 Besides provableness of what has been delivered / received when, for messages exchange patterns
1650 using the MsgBox service it may be of interest for the Initiator to be informed, when the intended
1651 Recipient pulls the message from his MsgBox. More concrete – the business scenario needs of an
1652 asynchronous message are bound to reaction times. In this case, a service requestor has to have
1653 control to in-time delivery to the targeted Recipient. In doubt there isn't any Recipient activity
1654 concerning the request, a service requestor (or even responder) may choose other communication
1655 channels to get in contact.

1656 As there may be non-conformant implementations which don't answer to a requested
1657 ReceptionReceipt, for additional comfort of control whether a message has been pulled yet by the
1658 intended Recipient, the construct of a **FetchNotification** is foreseen, which alike described for
1659 receipts can be demanded by Initiator and Recipient instances. If requested, the Recipients MsgBox
1660 instance MUST deliver such a notification to the endpoint the Initiator specified in his message;
1661 contents are the SOAP header elements indicating source and destination of the message and the
1662 time instant when it is pulled by the intended Recipient. No separate signature is foreseen for this
1663 notification. The FetchNotification is delivered in the SOAP body of a separate osci:Request to the
1664 endpoint to be exposed in the demand for this FetchNotification – which again in general should be
1665 the MsgBox of the requesting Initiator or Recipient node.

1666 8.3.1 Demanding Receipts

1667 To demand receipts and define its details, for each specific demand the here defined SOAP header
1668 blocks MAY be provided in outbound messages of type osci:Request and osci:Response by
1669 SOAP/OSCI endpoints (Initiator or Recipient).

1670 8.3.1.1 Demand for Delivery Receipt

1671 If the next logical OSCI node a message of type osci:Request is targeted to shall deliver a
1672 DeliveryReceipt in the backchannel osci:Response message, following SOAP header block MUST be
1673 included in the message:

```
1674 <osci:DeliveryReceiptDemand @wsu:Id="xs:ID"  
1675   @s12:role=  
1676     "http://www.w3.org/2003/05/soap-envelope/role/next" ?  
1677   @s12:mustUnderstand="true" | "false" ?  
1678   @qualTSPforReceipt="true" | "false" ?  
1679   @echoRequest="true" | "false") ? >  
1680   <wsa:ReplyTo> wsa:EndpointReference <wsa:ReplyTo>  
1681 </osci:DeliveryReceiptDemand> ?
```

1682 Description of elements and attributes in the schema overview above:

1683 **/osci:DeliveryReceiptDemand ?**

1684 Optional SOAP header for indicating requirements for a DeliveryReceipt. It MUST be
1685 provided under the conditions mentioned above.

1686 **/osci:DeliveryReceiptDemand/@wsu:Id**

1687 This attribute of type **wsu:Id** SHOULD be provided so that un-ambiguous references can
1688 be made to this element.

1689 **/osci:DeliveryReceiptDemand/@s12:role**

1690 This attribute of type **xs:anyURI** MAY be provided. It defaults to the URI outlined above.
1691 If this attribute is provided, it MUST be set to this value. Following the semantics of

- 1692 [SOAP12], this SOAP header block is designated to next SOAP-node passed on the
1693 message route.
- 1694 **/osci:DeliveryReceiptDemand/@s12:mustUnderstand**
- 1695 This boolean attribute SHOULD be provided with a value of "true". Following the
1696 semantics of [SOAP12], this SOAP header block MUST be understood and processed by
1697 the next SOAP-node passed on the message route willing to act in the role denoted by the
1698 foregoing attribute **/osci:ReceiptDemand/@s12:role**. For interoperability reasons
1699 with Web Service implementations not able to process the receipts defined here, it may be
1700 set to "false" or not present (which is equivalent to a value of "false").
- 1701 **/osci:DeliveryReceiptDemand/@qualTSPforReceipt ?**
- 1702 This optional boolean attribute signals – if set to a value of "true" – a qualified timestamp
1703 for the receipt information is requested. If such a service is not available on the node the
1704 receipt is demanded from, a fault (see chapter [8.3.2]) MUST be generated to the
1705 requesting node and the incoming message MUST be discarded.
- 1706 **/osci:DeliveryReceiptDemand/@echoRequest ?**
- 1707 This optional boolean attribute signals – if set to a value of "true" – the requesting node
1708 requires the retransmission of the whole message in the required receipt. In this case, the
1709 node the receipt is demanded from MUST provide the whole message in binary format in
1710 the receipt part of the response message (see chapter [8.3.2.1]). Care should be taken to
1711 use this feature with regard to caused overhead and bandwidth consumption.
- 1712 If absent, this attribute defaults to a value of "false".
- 1713 **/osci:DeliveryReceiptDemand/wsa:ReplyTo**
- 1714 This required element of type **wsa:EndpointReferenceType** denotes the endpoint,
1715 where the requestor wishes the receipt should be routed to. In case of a DeliveryReceipt
1716 demand in a message of type **osci:Request**, the value herein for **.../wsa:Address**
1717 SHOULD be **http://www.w3.org/2005/08/addressing/anonymous**; the
1718 DeliveryReceipt is returned directly in the header of the response to the incoming
1719 message in the same http-connection.
- 1720 In case the requestor wishes a DeliveryReceipt should be routed some specialized
1721 endpoint consuming receipts the EPR of the endpoint MUST be exposed here. The
1722 DeliveryReceipt is this case MUST be delivered in the SOAP body of a separate new
1723 **osci:Request** message. Hence, this EPR SHOULD be the one of the **MsgBox** instance of
1724 the requestor. It MAY even be a specialized endpoint consuming receipts. The EPR
1725 MUST contain reference properties according to chapter [6, Addressing Endpoints]. A
1726 **.../wsa:ReferenceParameters** of following value SHOULD be provided:
- 1727 **<osci:TypeOfBusinessScenario>**
1728 **www.osci.eu/2008/common/urn/messageTypes/Receipt**
1729 **</osci:TypeOfBusinessScenario>**.
- 1730 In case of delivering a receipt to a **MsgBox** instance, this is the defaulted value for
1731 separating receipt message types from other ones (see chapter [6, Addressing
1732 Endpoints]).
- 1733 A **/osci:DeliveryReceiptDemand** header block MUST NOT be included in an **osci:Response**
1734 message. As for an **osci:Response**, there is no network backchannel available; in this case
1735 DeliveryReceipt could not be delivered in the standard manner. If provability of response delivery is
1736 needed, an **/osci:ReceptionReceiptDemand** should be used instead.
- 1737 For synchronous request-response scenarios driven point-to-point between instances of initiator and
1738 recipient, it is advisable to economize demands for receipts to avoid overhead and processing of not
1739 needed DeliveryReceipts. Provability of communication here MAY be gained by a reception receipt

1740 requirement positioned in one or more messages of the request-response sequence, depending on
 1741 underlying concrete business process needs. Certainty of delivery itself is implicit given by successful
 1742 processing of such a scenario.

1743 **8.3.1.2 Demand for ReceptionReceipt**

1744 If an endpoint a message if type `osci:Request` or `osci:Response` is targeted to shall deliver a
 1745 `ReceptionReceipt`, following SOAP header block **MUST** be included in the message. The underlying
 1746 schema is the same as for an `osci:DeliveryReceiptDemand` SOAP header block; possible
 1747 attribute/element values and semantics differ in detail as described below.

```
1748 <osci:ReceptionReceiptDemand @wsu:Id="..."
1749   @s12:role=
1750     "http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver" ?
1751   @s12:mustUnderstand= "true" | "false" ?
1752   @qualTSPforReceipt="true" | "false" ?
1753   @echoRequest= "true" | "false") ? >
1754   <wsa:ReplyTo> wsa:EndpointReference <wsa:ReplyTo>
1755 </osci:ReceptionReceiptDemand> ?
```

1756 Description of elements and attributes in the schema overview above:

1757 `/osci:ReceptionReceiptDemand` ?

1758 Optional SOAP header for indicating requirements for a `ReceptionReceipt`.

1759 `/osci:ReceptionReceiptDemand/@wsu:Id`

1760 This attribute of type `wsu:Id` **SHOULD** be provided so that un-ambiguous references can
 1761 be made to this element.

1762 `/osci:ReceptionReceiptDemand/@s12:role`

1763 This attribute of type `xs:anyURI` **MAY** be provided. It defaults to the URI outlined above.
 1764 If this attribute is provided, it **MUST** be set to this value; following the semantics of
 1765 [SOAP12], this SOAP header block is designated SOAP-node acting in the role of a
 1766 ultimate receiver – which is the node at least the SOAP body is designated to,
 1767 corresponding to the `UltimateRecipient` node in the role model of this specification.

1768 `/osci:ReceptionReceiptDemand/@s12:mustUnderstand`

1769 This boolean attribute **SHOULD** be provided with a value of "true". Following the
 1770 semantics of [SOAP12], this SOAP header block **MUST** be understood and processed by
 1771 the next SOAP-node passed on the message route willing to act in the role denoted by the
 1772 foregoing attribute `/osci:ReceptionReceiptDemand/@s12:role`. For interoperability reasons
 1773 with Web Service implementations not able to process the receipts defined here, it may be
 1774 set to "false" or not present (which is equivalent to a value of "false").

1775 `/osci:ReceptionReceiptDemand/@qualTSPforReceipt` ?

1776 This optional boolean attribute signals – if set to a value of "true" – a qualified timestamp
 1777 for the receipt information is requested. If such a service is not available on the node the
 1778 receipt is demanded from, a fault (see chapter [8.3.2]) **MUST** be generated to the
 1779 requesting node and the message **MUST** be discarded.

1780 `/osci:ReceptionReceiptDemand/@echoRequest` ?

1781 This optional boolean attribute signals – if set to a value of "true" – the requesting node
 1782 requires the retransmission of the whole message in the required receipt. In this case, the
 1783 node the receipt is demanded from **MUST** provide the whole message in binary format in
 1784 the receipt part of the response message (see chapter [8.3.2.2]). Care should be taken to
 1785 use this feature with regard to caused overhead and bandwidth consumption.

1786 If absent, this attribute defaults to a value of "false".

1787 `/osci:ReceptionReceiptTo/wsa:ReplyTo`

1788 This required element of type `wsa:EndpointReferenceType` denotes the endpoint,
 1789 where the requestor wishes the receipt should be routed to. A `ReceptionReceipt` in
 1790 general MUST be delivered in the SOAP body of a separate new `osci:Request` message,
 1791 hence this EPR SHOULD be the one of the `MsgBox` instance of the requestor or MAY be
 1792 a specialized endpoint consuming receipts. The EPR MUST contain reference properties
 1793 according to chapter [6, Addressing Endpoints]. A `.../wsa:ReferenceParameters` of
 1794 following value SHOULD be provided:

```
1795 <osci:TypeOfBusinessScenario>
1796     www.osci.eu/2008/common/urn/messageTypes/Receipt
1797 </osci:TypeOfBusinessScenario>
```

1798 In case off delivering a receipt to a `MsgBox` instance, this is the defaulted value for
 1799 separating receipt message types from other ones (see chapter [6, Addressing
 1800 Endpoints]).

1801 8.3.2 Receipt Format and Processing

1802 8.3.2.1 Delivery Receipt

1803 `DeliveryReceipts` MUST be produced immediately after successful acceptance of an incoming
 1804 message of type `osci:Request`, if a SOAP header element `/osci:DeliveryReceiptDemand` is
 1805 present in the incoming `osci:Request` message.

1806 The data for this type of receipt has to be carried in the resulting SOAP response message in following
 1807 SOAP header block `/osci:DeliveryReceipt`:

```
1808 <osci:DeliveryReceipt @wsu:Id="xs:ID"
1809   <osci:ReceiptInfo
1810     @wsu:Id="..."
1811     @osci:ReceiptIssuerRole=
1812       "http://www.osci.eu/ws/2008/05/transport/role/MsgBox" |
1813       "http://www.osci.eu/ws/2008/05/transport/role/Recipient"
1814
1815     <wsa:MessageID>xs:anyURI</wsa:MessageID>
1816     <osci:MsgTimeStamps/>
1817     <wsa:RelatesTo/> *
1818     <osci:To> EPR </osci:To>
1819     <wsa:ReplyTo> EPR </wsa:ReplyTo>
1820     <wsa:From> EPR </wsa:From> ?
1821     <osci:RequestEcho> xs:base64Binary </RequestEcho> ?
1822   </osci:ReceiptInfo>
1823   <ds:Signature/>
1824 </osci:Receipt>
```

1825 Description of elements and attributes in the schema overview above:

1826 `/osci:DeliveryReceipt`

1827 Container holding the child elements receipt data `.../osci:ReceiptInfo` and a
 1828 `ds:Signature` element over `.../osci:ReceiptInfo`.

1829 `/osci:DeliveryReceipt/@wsu:Id`

1830 This attribute of type `xs:ID` MUST be provided so that un-ambiguous references (i.e. for
 1831 transport signature and encryption) can be made to this `/osci:DeliveryReceipt`
 1832 block.

1833 `/osci:DeliveryReceipt/osci:ReceiptInfo`

1834 Container to hold the receipt details.

- 1835 **/osci:DeliveryReceipt/osci:ReceiptInfo/wsua:Id**
- 1836 This attribute of type **xs:ID** MUST be provided; the element must be referenceable from
1837 the signature element described below.
- 1838 **/osci:DeliveryReceipt/osci:ReceiptInfo/osci:ReceiptIssuerRole**
- 1839 This element of type **xs:anyURI** MUST be provided with one of the URIs outlined above.
1840 The concrete value MUST expose the role of the receipt issuing node. If an **osci:Request**
1841 is targeted to a **MsgBox** instance, the value MUST be
1842 "**http://www.osci.eu/ws/2008/05/transport/role/MsgBox**". If an
1843 **osci:Request** if targeted directly to the recipients OSCI Gateway or an **osci:Response**
1844 message contains a demand for a **DeliveryReceipt**, the value MUST be
1845 "**http://www.osci.eu/ws/2008/05/transport/role/Recipient**".
- 1846 **/osci:DeliveryReceipt/osci:ReceiptInfo/wsa:MessageID**
- 1847 The **/wsa:MessageID** SOAP header block of the message to be receipted.
- 1848 **/osci:ReceptionReceipt/osci:ReceiptInfo/osci:MsgTimeStamps**
- 1849 The **/osci:MsgTimeStamps** SOAP header block; this element MUST be inserted after
1850 the receipting node has inserted his specific timestamps according to chapter [8.1].
- 1851 **/osci:DeliveryReceipt/osci:ReceiptInfo/wsa:RelatesTo ***
- 1852 The **/wsa:RelatesTo** SOAP header blocks of the message to be receipted.
- 1853 **/osci:DeliveryReceipt/osci:ReceiptInfo/osci:To**
- 1854 This element of type **wsa:EndpointReference** denotes the destination EPR of the
1855 message to be receipted. At least, it MUST contain the **/wsa:To** SOAP header block of
1856 this message and those SOAP header blocks attributed by
1857 **@wsa:IsReferenceParameter="1"**.
- 1858 **/osci:DeliveryReceipt/osci:ReceiptInfo/wsa:From ?**
- 1859 If present in the message to be receipted, the **/wsa:From** SOAP header block of the
1860 message to be receipted.
- 1861 **/osci:DeliveryReceipt/osci:ReceiptInfo/wsa:ReplyTo**
- 1862 The **/wsa:From** SOAP header block of the message to be receipted.
- 1863 **/osci:DeliveryReceipt/osci:RequestEcho ?**
- 1864 This element MUST be included, if the demand for the receipt contains the attribute
1865 **/osci:DeliveryReceiptDemand/@echoRequest** set to a value of "**true**". The
1866 complete incoming message MUST be placed in this element in **base64Binary** format.
1867 To be able to proof what has been sent, the Initiator in this case is strongly advised to
1868 encrypt the message body for himself, too.
- 1869 **/osci:DeliveryReceipt/ds:Signature**
- 1870 A digital signature of the **DeliveryReceipt** according to chapter [7.2].
- 1871 One **ds:Signature** child element **ds:Reference** MUST point to the element
1872 **/osci:DeliveryReceipt/ReceiptInfo** using the same-URI reference mechanism
1873 via the ID-attribute of this element.
- 1874 A second **/ds:Signature/ds:Reference** element MUST point to the
1875 **/s12:Envelope/s12:Body** block of the message to be receipted using the same-URI
1876 reference mechanism via the ID-attribute of the SOAP body block.
- 1877 For a **DeliveryReceipt**, the received SOAP body block MUST to be signed "as is". The
1878 actual server time in UTC-format MUST be provided in


```

1879 /osci:DeliveryReceipt/ds:Signature/ds:Object/
1880 xades:QualifyingProperties/xades:SignedProperties/
1881 xades:SignedSignatureProperties/xades/SigningTime.

```

1882 If in the receipt demand SOAP header actually processed, the attribute
1883 /osci:DeliveryReceiptDemand/@qualTSPforReceipt is set to a value of
1884 "true" and can be served from this instance, the signature element MUST be extended
1885 by a *qualified timestamp over the signature itself*. For the timestamp itself, the
1886 specification [RFC3161] applies, the placement in the signature element follows [XAdES]
1887 as described in chapter [7.2.2]:

```

1888 ../ds:Signature/ds:Object/xades:QualifyingProperties/
1889 xades:UnsignedProperties/xades:UnsignedSignatureProperties/
1890 xades:SignatureTimeStamp/xades:EncapsulatedTimeStamp

```

1891 If no appropriate qualified TSP-service can be provided, a fault MUST be generated to the
1892 requestor and processing of the incoming message MUST be aborted.

1893	Fault 8: QualTSPServiceNotAvailable
1894	[Code] Sender
1895	[Subcode] QualTSPServiceNotAvailable
1896	[Reason] Requested qualified TSP service not provided by targeted node
1897	The fault [Details] property MUST outline that this timestamp was requested for a
1898	DeliveryReceipt and that the message is not accepted.

1899 If an incoming message of type osci:Request is to be receipted, the block /osci:DeliveryReceipt
1900 MUST be included as SOAP header in the corresponding osci:Response message.

1901 If the message to be receipted is of type osci:Response, the block /osci:DeliveryReceipt MUST
1902 be positioned as SOAP body of a new osci:Request message. This osci:Request message MUST be
1903 targeted to the endpoint denoted in /osci:DeliveryReceiptDemand/wsa:ReplyTo. The SOAP
1904 header block /wsa:RelatesTo of this message MUST be supplied with the /wsa:MessageID
1905 SOAP header block of the message to be receipted.

1906 8.3.2.2 Reception Receipt

1907 If demanded by a osci:ReceptionReceiptDemand SOAP header of a osci:Request or
1908 osci:Response message, Reception Receipts MUST be processed after successful decryption of the
1909 SOAP body block. Depending on the concrete arrangement of roles in an OSCI endpoint
1910 implementation it may be possible that decryption of the SOAP body and processing of a
1911 ReceptionReceipt demand is decoupled from the node that accepts incoming requests respective
1912 responses (where DeliveryReceipt demands have to be processed immediately). Thus, a
1913 ReceptionReceipt is generated by Ultimate Recipient instances. For a message of type
1914 osci:Response, this is the Ultimate Recipient instance on the Initiator side.
1915 The data for this type of receipt has to be placed into following block /osci:ReceptionReceipt by
1916 the receipt generating node. The underlying schema nearly the same as for a
1917 osci:DeliveryReceipt SOAP header block; possible attribute/element values and semantics
1918 differ in detail as described here.

```

1919 <osci:ReceptionReceipt @wsu:Id="xs:ID" ? >
1920 <osci:ReceiptInfo @wsu:Id="..." >
1921
1922 <wsa:MessageID>xs:anyURI</wsa:MessageID>
1923 <osci:MsgTimeStamps/>
1924 <wsa:RelatesTo/> *
1925 <osci:To> EPR </osci:To>
1926 <wsa:ReplyTo> EPR </wsa:ReplyTo>
1927 <wsa:From> EPR </wsa:From> ?
1928 <osci:RequestEcho> xs:base64Binary </RequestEcho> ?
1929 </osci:ReceiptInfo>

```

1930	<code><ds:Signature/></code>
1931	<code></osci:Receipt></code>
1932	Description of elements and attributes in the schema overview above:
1933	<code>/osci:ReceptionReceipt</code>
1934	Container holding the child elements receipt data <code>.../osci:ReceiptInfo</code> and a
1935	<code>ds:Signature</code> element over <code>.../osci:ReceiptInfo</code> .
1936	<code>/osci:ReceptionReceipt/@wsu:Id</code>
1937	This attribute of type <code>xs:ID</code> SHOULD be provided so that un-ambiguous can be made to
1938	this <code>/osci:ReceptionReceipt</code> block.
1939	<code>/osci:ReceptionReceipt/osci:ReceiptInfo</code>
1940	Container to hold the receipt details.
1941	<code>/osci:ReceptionReceipt/osci:ReceiptInfo/wsu:@Id</code>
1942	This attribute of type <code>xs:ID</code> MUST be provided; the element must be referenceable from
1943	the signature element described below.
1944	<code>/osci:ReceptionReceipt/osci:ReceiptInfo/wsa:MessageID</code>
1945	The <code>/wsa:MessageID</code> SOAP header block of the message to be received.
1946	<code>/osci:ReceptionReceipt/osci:ReceiptInfo/osci:MsgTimeStamps</code>
1947	The <code>/osci:MsgTimeStamps</code> SOAP header block; this element MUST be inserted after
1948	the receiving node has inserted his specific timestamps according to chapter [8.1].
1949	<code>/osci:ReceptionReceipt/osci:ReceiptInfo/wsa:RelatesTo *</code>
1950	The <code>/wsa:RelatesTo</code> SOAP header blocks of the message to be received.
1951	<code>/osci:ReceptionReceipt/osci:ReceiptInfo/osci:To</code>
1952	This element of type <code>wsa:EndpointReference</code> denotes the destination EPR of the
1953	message to be received. At least, it MUST contain the <code>/wsa:To</code> SOAP header block of
1954	this message and those SOAP header blocks attributed by
1955	<code>@wsa:IsReferenceParameter="1"</code> .
1956	<code>/osci:ReceptionReceipt/osci:ReceiptInfo/wsa:From ?</code>
1957	If present in the message to be received, the <code>/wsa:From</code> SOAP header block of the
1958	message to be received.
1959	<code>/osci:ReceptionReceipt/osci:ReceiptInfo/wsa:ReplyTo</code>
1960	The <code>/wsa:ReplyTo</code> SOAP header block of the message to be received.
1961	<code>/osci:ReceptionReceipt/osci:RequestEcho ?</code>
1962	This element MUST be included, if the demand for the receipt contains the attribute
1963	<code>/osci:ReceptionReceiptDemand/@echoRequest</code> set to a value of <code>"true"</code> . The
1964	complete incoming message MUST be placed in this element in base64Binary format.
1965	<code>/osci:ReceptionReceipt/ds:Signature</code>
1966	A digital signature of the ReceptionReceipt according to chapter [7.2.2]. The signature
1967	MUST be generated by the Ultimate Recipient after successful decryption of the whole
1968	SOAP body block. In case a synchronous <code>osci:Response</code> to an <code>osci:Request</code> containing a
1969	ReceptionReceipt demand, this is the respective UltimateRecipient node on the Initiator
1970	side. If complete decryption of the received SOAP body is not possible, a fault MUST be
1971	generated and further message processing MUST be aborted.
1972	Fault 9: MsgBodyDecryptionError

1973	[Code] Sender
1974	[Subcode] MsgBodyDecryptionError
1975	[Reason] Message body decryption failed.
1976	The fault [Details] property MAY outline – if known to the decrypting instance - the
1977	<code>ds:X509IssuerSerial</code> element of the certificate initially used for encryption.
1978	One <code>ds:Signature</code> child element <code>ds:Reference</code> MUST point to the element
1979	<code>/osci:ReceptionReceipt/ReceiptInfo</code> using the same-URI reference mechanism
1980	via the ID-attribute of this element.
1981	A second <code>/ds:Signature/ds:Reference</code> element MUST point to the element
1982	<code>/s12:Envelope/s12:Body</code> element of the message to be receipted using the same-
1983	URI reference mechanism via the ID-attribute of the SOAP body block. As already
1984	mentioned, the SOAP body block in advance MUST have been successfully decrypted.
1985	The actual server time in UTC-format MUST be provided in the child element
1986	<code>/xades:SigningTime</code> of <code>/osci:ReceptionReceipt/ds:Signature/ds:Object/</code>
1987	<code>xades:QualifyingProperties/xades:SignedProperties/</code>
1988	<code>xades:SignedSignatureProperties</code> .
1989	If in the receipt demand SOAP header actually processed, the attribute
1990	<code>/osci:ReceptionReceiptDemand/@qualTSPforReceipt</code> is set to a value of
1991	"true" and can be served from this instance, the signature element MUST be extended
1992	by a <i>qualified timestamp over the signature itself</i> . For the timestamp itself, the
1993	specification [RFC3161] applies, the placement in the signature element follows [XAdES]
1994	as described in chapter [7.2.2]:
1995	<code>.../ds:Signature/ds:Object/xades:QualifyingProperties/</code>
1996	<code>xades:UnsignedProperties/</code>
1997	<code>xades:UnsignedSignatureProperties/</code>
1998	<code>xades:SignatureTimeStamp/xades:EncapsulatedTimeStamp</code>
1999	If no appropriate qualified TSP-service can be provided, a fault MUST be generated to the
2000	requestor instead of the ReceptionReceipt, processing of the incoming message MAY
2001	proceed (subject to policy to be defined for the concrete endpoint). See fault
2002	QualTSPServiceNotAvailable as defined in chapter [8.3.2.1]; the fault [Details] property
2003	MUST outline that this timestamp was requested for a ReceptionReceipt and if further
2004	message processing takes place or not.
2005	The block <code>/osci:ReceptionReceipt</code> MUST be positioned as SOAP body of a new <code>osci:Request</code>
2006	message. This <code>osci:Request</code> message MUST be targeted to the endpoint denoted in
2007	<code>/osci:ReceptionReceiptDemand/wsa:ReplyTo</code> . The SOAP header block <code>/wsa:RelatesTo</code> of
2008	this message MUST be supplied with the <code>/wsa:MessageID</code> SOAP header block of the message to
2009	be receipted.

2010 **8.3.2.3 Additional Receipt/Notification Demand fault processing Rules**

2011 As fault occurrence is imaginable while processing receipt demands, it must be foreseen to
 2012 communicate those faults to the requestor of a receipt. As far as a receipt has to be delivered directly
 2013 in the SOAP header of a response to a request in the same http-connection, such a fault occurrence is
 2014 directly communicated to the requestor by the general SOAP/OSCI fault processing mechanisms and
 2015 the message is discarded. **No receipt SOAP header block MUST be build up and inserted in the**
 2016 **response in this case.**

2017 For receipts which have to be processed asynchronous mode and/or delivered in a separate
 2018 `osci:Request` message, the receipt requestor has to be informed about possible fault occurrence
 2019 asynchronously. This MUST be done by placing the fault information in the body of an `osci:Request`
 2020 instead of the receipt block and delivered to the same endpoint the receipt message is expected.

2021 If the message to be receipted carries a `wsa:FaultTo` SOAP header block, this is the EPR the
 2022 `osci:Request` message carrying the fault MUST be targeted to. If this header is absent or – if present
 2023 and carrying a value of `http://www.w3.org/2005/08/addressing/anonymous` in
 2024 `wsa:FaultTo/Address`, it MUST be targeted to the endpoint denoted in
 2025 `/osci:ReceptionReceiptDemand/wsa:ReplyTo/Address`. The according
 2026 `wsa:ReferenceParameters` SOAP header block MUST be

```
2027 <osci:TypeOfBusinessScenario wsa:IsReferenceParameter="true">
2028     www.osci.eu/2008/common/urn/messageTypes/Fault
2029 </osci:TypeOfBusinessScenario>
```

2030 The SOAP header block `/wsa:RelatesTo` of this message MUST be supplied with the
 2031 `/wsa:MessageID` SOAP header block of the message to be receipted.

2032 ReceptionReceipts and FetchedNotification in general have to be delivered in asynchronous mode. If
 2033 the request for these doesn't indicate a valid address they can be successfully targeted to, standard
 2034 SOAP addressing error handling applies. It is strongly advised, that these errors SHOULD be logged
 2035 at the node producing the requested receipt/notification; follow-up of this situation is up to the
 2036 operating policies¹⁹.

2037 **8.3.2.4 Receipt Signature Validation**

2038 Receipt signatures MUST be verified by receipt consuming nodes. If signature verification fails, a fault
 2039 MUST be generated and be made available to the Source Application instance initially triggering the
 2040 corresponding receipt demand. Fault delivery in this case is a matter of implementation.

2041	Fault 10: SignatureOfReceiptInvalid
2042	[Code] Sender
2043	[Subcode] SignatureOfReceiptInvalid
2044	[Reason] Receipt signature verification failed.

2045 The fault [Details] property SHOULD outline the concrete verification failure. It is on behalf of the
 2046 Source Application to handle this situation.

2047 **8.3.3 Fetched Notification**

2048 To demand a FetchedNotification from a recipient MsgBox instance where the message is relayed,
 2049 following SOAP header block MUST be provided in an `osci:Request` message:

```
2050 <osci:FetchedNotificationDemand @wsu:Id="..." ?
2051     @s12:role="http://www.osci.eu/ws/2008/05/transport/role/MsgBox" >
2052     <wsa:ReplyTo>wsa:EndpointReference</wsa:replyTo>
2053 </osci:FetchedNotificationDemand>
```

2054 Description of elements and attributes in the schema overview above:

2055 `/osci:FetchedNotificationDemand`

2056 Header block contain the demand.

2057 `/osci:FetchedNotificationDemand/@wsu:Id`

2058 For ease of referencing this SOAP header block from WS Security SOAP header
 2059 elements, this attribute of type `wsu:Id` SHOULD be provided.

2060 `/osci:FetchedNotificationDemand/@s12:role`

¹⁹ Those should be made available online for all possible communication partners. Details are not addressed by this document.

2061 This attribute of type **xs:anyURI** MUST be provided with the URI outlined above. Only
 2062 nodes acting in the role **MsgBox** are addressed by this type of demand.

2063 **/osci:ReceiptTo/wsa:ReplyTo**

2064 This required element of type **wsa:EndpointReferenceType** denotes the endpoint,
 2065 where the requestor wishes the notification should be routed to. As **FetchNotifications**
 2066 can only be delivered in the SOAP body of a separate new message, this EPR SHOULD
 2067 be the one of the **MsgBox** instance of the requestor or MAY be a specialized endpoint
 2068 consuming notifications. The EPR MUST contain reference properties according to
 2069 chapter [6, Addressing Endpoints]. A **.../wsa:ReferenceParameters** of following value
 2070 SHOULD be provided:

2071 **<osci:TypeOfBusinessScenario>www.osci.eu/2008/common/urn/messageTypes/Notification/>**. In case of delivering a receipt to a **MsgBox** instance, this is the
 2072 defaulted value for separating notification message types from other ones (see chapter [6,
 2073 Addressing Endpoints]).
 2074

2075 A SOAP header **/osci:FetchNotificationDemand** MUST be processed by a node acting in
 2076 the role of **MsgBox** when the message is pulled the first time by the Recipient of the message. It
 2077 MUST be delivered in a separate message to the endpoint denoted in the appropriate demand. They
 2078 MUST only be produced by **MsgBox** instances. The **/osci:FetchNotification** block is
 2079 positioned in the body of such a message, other body parts MUST NOT be included.

2080 Syntax for messages to deliver **FetchNotifications**:

```

2081 <s12:Envelope ...>
2082   <s12:Header ...>
2083     ...
2084     <wsa:Action>
2085       www.osci.eu/2008/transport/urn/messageTypes/Notification
2086     </wsa:Action>
2087     <wsa:MessageID>xs:anyURI</wsa:MessageID>
2088     <wsa:RelatesTo>xs:anyURI</wsa:RelatesTo>
2089     <wsa:To>xs:anyURI</wsa:To>
2090     <osci:TypeOfBusinessScenario @wsa:IsReferenceParameter="1">
2091       "www.osci.eu/2008/common/urn/messageTypes/Notification"/>
2092     </osci:TypeOfBusinessScenario>
2093     ...
2094   </s12:Header>
2095   <s12:Body>
2096     <osci:FetchNotification @wsu:Id="..." ?>
2097       <osci:FetchTime> xs:dateTime </osci:FetchTime>
2098       <wsa:MessageID>xs:anyURI</wsa:MessageID>
2099       <wsa:To> esa:Address </wsa:To>
2100       <wsa:From> EPR </wsa:From>
2101     </osci:FetchNotification>
2102   </s12:Body>
2103 </s12:Envelope>
  
```

2104 Description of elements and attributes in the schema overview above:

2105 **/s12:Envelope/s12:Header/wsa:Action**

2106 The value indicated herein MUST be used for that URI.

2107 **/s12:Envelope/s12:Header/wsa:MessageID**

2108 The message MUST carry a unique WS-Addressing MessageID.

2109 **/s12:Envelope/s12:Header/wsa:RelatesTo**

2110 The message MUST carry the WS-Addressing MessageID for the message a
 2111 **FetchNotification** was requested for.

2112 **/s12:Envelope/s12:Header/wsa:To**

- 2113 The address of the destination endpoint which was stated in the EPR of the request
 2114 header element `/osci:FetchNotificationTo/wsa:ReplyTo/wsa:Address` of
 2115 the request message.
- 2116 `/s12:Envelope/s12:Header/osci:TypeOfBusinessScenario`
- 2117 This is the instantiation of `/wsa:ReferenceParameters` bound to this EPR. It MUST be
 2118 taken from the request header element
 2119 `/osci:FetchNotificationTo/wsa:ReplyTo/wsa:ReferenceParameters` of
 2120 the request message.
- 2121 `/s12:Envelope/s12:Header/osci:TypeOfBusinessScenario/`
 2122 `@wsa:IsReferenceParameter`
- 2123 Following WS-Addressing, the element MUST be attributed with
 2124 `@wsa:IsReferenceParameter="1"`
- 2125 `/s12:Envelope/s12:Body/osci:FetchNotification`
- 2126 Container holding the `FetchNotification`.
- 2127 `/osci:Receipt/osci:FetchNotification/osci:FetchTime`
- 2128 This element of type `xs:dateTime` MUST be set to the actual server time instance in
 2129 UTC format when the `MsgBox` node processes the `FetchNotification` demand (message
 2130 first time pulled from recipient).
- 2131 `/s12:Envelope/s12:Body/osci:FetchNotification/wsa:MessageID`
- 2132 The `.../wsa:MessageID` of the message to give a `FetchNotification` for.
- 2133 `/s12:Envelope/s12:Body/osci:FetchNotification/wsa:To`
- 2134 The `.../wsa:To` element of the message to give a `FetchNotification` for.
- 2135 `/s12:Envelope/s12:Body/osci:FetchNotification/wsa:From`
- 2136 The `.../wsa:From` header element of the message to give a `FetchNotification` for.

2137 **8.4 X.509-Token Validation on the Message Route**

- 2138 A custom SOAP header is defined here for carrying X.509-Certificates and details per usage instance
 2139 in the referred message body block parts.
- 2140 Certificate validation processing SOAP nodes MUST enrich the message SOAP header block with the
 2141 gathered certificate validation results and – for processing optimization purposes – mark those usage
 2142 instances of a certificate as "checked", if validation could be processed successfully.
- 2143 An XML-Syntax to carry validation results is defined by XKMS 2/XKISS [XKMS], which is incorporated
 2144 here. The `/xkms:ValidateResult` specified in XKMS includes original validation responses from
 2145 CAs like OCSP-Responses and CRLs. In addition to [XKMS], extensions are defined to satisfy
 2146 requirements coming out the German signature law/directive regarding certificate validation. These
 2147 extensions are optional in general, but MUST be provided from OSCI service providers in Germany.
- 2148 Ultimate Recipients of messages MAY rely on the validation information thus once included in the
 2149 message body. As at least the inner CA-Responses are verifiable, as they are carrying signatures of
 2150 respective validation responders (OCSP, CRL...). In general, it's up to each node or endpoint on the
 2151 message route to rely on the validation information found in the message or to initiate revalidation of
 2152 used certificates following own needs und trust relations.
- 2153 This specification enforces no rules how a node serving certificate validation obtains certificate
 2154 validation results. It SHOULD be preferred to use the services of a trusted XKMS/XKISS responder
 2155 instance, like this a `/xkms:ValidateResult` can easily be gathered by the corresponding
 2156 `/xkms:ValidateRequest`. If the used XKMS/XKISS responder is designed as a relay bridging links

2157 to all relevant CAs concerning the overall requirements of a concrete OSCI based communication
 2158 network , the burden of administrating CA-links and serving further protocols for those links is
 2159 delegated to the XKMS/XKISS responder provider.

2160 If using a XKMS responder, it is advisable to use the advantage of compound validation request
 2161 offered by the XKMS/XKISS protocol. All validation requests for all usage instances of the certificates
 2162 exposed in the `/osci:X509TokenContainer` custom SOAP header block MAY be combined in one
 2163 compound request, which leads to a corresponding compound response. See [XKMS] for further
 2164 details.

2165 **8.4.1 X.509-Token Container**

2166 This chapter describes this optional custom header to carry those certificates. In addition to the token
 2167 themselves, following information is carried, which has to be provided by Source- / Target Applications
 2168 per token-usage:

- 2169 • Where a certificate is used in the body (by IDREF); information may be useful at recipient
 2170 side when parsing a message after SOAP body block decryption and grouping together
 2171 derived body block parts with their respective certificates/validation results (at least,
 2172 validation of signatures contained in the body SHOULD happen now)
- 2173 • Application time instant (this is the time instant a certificate must be proven as valid)
- 2174 • Possibility to indicate forced online OCSP request to downstream validation service nodes
 2175 (force bypassing possible caching of once gained OCSP responses).

2176 While processing the validation, such a node supplies following additional information:

- 2177 • A reference to the corresponding `/xkms:ValidateResult` per usage instance
- 2178 • An indicator "validated" if all usage instance of a token have successfully been validated
 2179 (note: only indication the fact of validation, not the result!)
- 2180 • An indicator "validation completed" when all usage instances of all carried token have
 2181 successfully been validated.

2182 As under certain circumstances it may be, that a certificate validations serving node is not able to
 2183 gather all needed `/xkms:ValidateResult(s)` completely, the latter two indicators only serve for
 2184 processing optimization – they can be used to avoid iterating through the X509 token container and
 2185 checking for outstanding `/xkms:ValidateResult(s)` by downstream nodes / endpoints on the
 2186 message route.

2187 Syntax for an optional `/osci:X509TokenContainer`:

```
2188 <osci:X509TokenContainer validateCompleted=("true" | "false")? >
2189   <osci:X509TokenInfo Id="xs:ID"
2190     validated=("true" | "false")? >
2191     <ds:X509Data>
2192       <ds:X509Certificate>
2193     </ds:X509Data>
2194     <osci:TokenApplication
2195       ocsNoCache=("true" | "false")? >
2196       validateResultRef="xs:IDREF" ? >
2197       <osci:TimeInstant>xs:dateTime</osci:TimeInstant>
2198       <osci:MsgItemRef>xs:IDREF</osci:MsgItemRef>
2199     </osci:TokenApplication> +
2200   </osci:X509TokenInfo> +
2201 </osci:X509TokenContainer>
```

2202 Description of elements and attributes in the schema overview above:

2203 `/osci:X509TokenContainer` ?

2204 Optional SOAP header block containing the X.509-Certificates which SHOULD be
 2205 validated by a node on the message route with validation capabilities.

- 2206 This container SHOULD be provided by a Source Application together with the payload to
 2207 be placed in the message body block at OSCI gateway entry point towards applications.
 2208 If present in an incoming message, it MUST be provided to the addressed Target
 2209 Application by the recipients OSCI gateway.
- 2210 **/osci:X509TokenContainer/@validateCompleted ?**
- 2211 This optional boolean attribute MUST be provided with a value of "true" by a validation
 2212 processing node, when processing was successfully passed for all application instances
 2213 of all contained items **/osci:X509TokenInfo**. It MUST NOT provided with a value of
 2214 "true" if this condition is false – i.e. only partially successful validation processing. It MUST
 2215 NOT be provided or changed by other logical instances than validation processing nodes.
- 2216 **/osci:X509TokenContainer/osci:X509TokenInfo +**
- 2217 If an **/osci:X509TokenContainer** is present, it MUST contain at least one item of this
 2218 type; content description follows here:
- 2219 **/osci:X509TokenContainer/osci:X509TokenInfo/@Id**
- 2220 This mandatory attribute of type **xs:ID** MUST be provided. As the whole
 2221 **/osci:X509TokenContainer** is initially to be generated by a Source Application
 2222 instance, the value must be a UUID; the UUID-value MUST NOT start with a character
 2223 unlike the **xs:ID** production rules and SHOULD therefore be preceded by a string of
 2224 "uuid:". This attribute MUST NOT be provided or changed by other logical instances than
 2225 Source Applications.
- 2226 **/osci:X509TokenContainer/osci:X509TokenInfo/@validated ?**
- 2227 This optional boolean attribute of type **xs:ID** MUST be provided with a value of "true" by
 2228 a validation processing node, when processing was successfully passed for all application
 2229 instances of this item **/osci:X509TokenInfo**. It MUST NOT provided with a value of
 2230 "true" if this condition is false – i.e. only partially successful validation processing. It MUST
 2231 NOT be provided or changed by other logical instances than validation processing nodes.
- 2232 **/osci:X509TokenContainer/osci:X509TokenInfo/ds:X509Data**
- 2233 The X.509-Token of type **ds:Data** MUST be provided here by the Source Application
 2234 instance. Other sub-elements than **X509Certificate** foreseen in **ds:X509Data** MUST
 2235 NOT be provided.
- 2236 **/osci:X509TokenContainer/osci:X509TokenInfo/ds:X509Data/ds:X509Certificate**
- 2237 Sub-element **.../ds:509Certificate** MUST be provided. It MUST NOT be provided or
 2238 changed by other logical instances than Source Applications.
- 2239 **/osci:X509TokenContainer/osci:X509TokenInfo/osci:TokenApplication**
- 2240 A Source Application MUST initially provide this container containing application details of
 2241 the X.509-Token.
- 2242 **/osci:X509TokenContainer/osci:X509TokenInfo/osci:TokenApplication/
 2243 @ocspNoCache ?**
- 2244 This optional boolean attribute MUST be provided with a value of "true" by the Source
 2245 Application instance, when the downstream validation service node shall be forced
 2246 bypassing possible cacheing of OCSP responses while validating this certificate. If not
 2247 provided with a value of "true", a validation service node MAY use cacheing mechanisms
 2248 to build up validation results. It MUST NOT be provided or changed by other logical
 2249 instances than a Source Application instance.
- 2250 **/osci:X509TokenContainer/osci:X509TokenInfo/osci:TokenApplication/
 2251 @validateResultRef ?**

2252 Validation processing nodes MUST provide an `xs:IDREF` here when processing was
 2253 successfully passed for this instance of `/osci:X509TokenInfo/TokenApplication`.
 2254 It must point to the related `/xkms:ValidateResult` header child element (see next
 2255 chapter). It MUST NOT be provided or changed by other logical instances than validation
 2256 processing nodes. If present, this attribute indicates, that this instance of
 2257 `/osci:X509TokenInfo/osci:TokenApplication` is validated.

2258 `/osci:X509TokenContainer/osci:X509TokenInfo/osci:TokenApplication/
 2259 osci:TimeInstant`

2260 This element of type `xs:dateTime` MUST be provided by the Source Application
 2261 instance and carry the token application time instant. This time instant MUST be taken as
 2262 validation time instant by the validation processing node (see next chapter). It MUST NOT
 2263 be provided or changed by other logical instances than Source Applications.

2264 `/osci:X509TokenContainer/osci:X509TokenInfo/osci:TokenApplication/
 2265 osci:MsgItemId`

2266 This element of type `xs:IDREF` MUST be provided by the Source Application instance
 2267 and carry a reference to the cryptographic element in the message body where the token
 2268 was used. It MUST NOT be provided or changed by other logical instances than Source
 2269 Applications.

2270 8.4.2 X.509-Token Validation Results

2271 SOAP nodes which are willing and able processing validation for X.509-Certificates contained in the
 2272 `/osci:X509TokenContainer` SOAP header block MUST insert the processing result in SOAP
 2273 header blocks `/xkms:CompoundResult` containing one or more `/xkms:ValidateResult`
 2274 elements conformant to the part XKISS of the XKMS specification. See [XKMS] for details, whereby
 2275 following profiling applies here:

2276 **R1100:** Validation results MUST be signed by the generating instance. This MAY be a XKMS-
 2277 Responders involved or – if no dedicated XKMS-Responder is used – the node generating
 2278 the header block `/xkms:CompoundResult` containing the `/xkms:ValidateResult`
 2279 elements. Hence, the element `/xkms:CompoundResult/ds:Signature` MUST be
 2280 present. The subordinate signature elements `/xkms:ValidateResult/ds:Signature`
 2281 SHOULD be omitted.

2282 **R1120:** For nodes consuming the validation results, it MUST be able to establish trust to the
 2283 validation results generating node through the certificate used for this signature. If no trust
 2284 can be established, these nodes MUST ignore the affected header block
 2285 `/xkms:CompoundResult` and MUST revalidate the affected certificates using a service
 2286 trusted by this node.

2287 **R1130:** Nodes consuming the validation results MUST validate the signature of the
 2288 `/xkms:CompoundResult`. If signature validation fails, this fact MUST be logged as a
 2289 security error including the affected header block. This header block MUST be ignored,
 2290 the affected certificates using a service trusted by this node.

2291 For XKMS messages an abstract extension point `xkms:MessageExtension` is foreseen to carry
 2292 additional information. German regulations as well as EU-wide efforts for alignment of interoperable
 2293 use of electronic signatures require detailed information on certificate quality, validity status, used
 2294 algorithm suitability and the validation process itself. Thus, a `/xkms:ValidateResult` SHOULD
 2295 contain an extension block `/xkmsEU`, XML namespace
 2296 `http://www.lsp.eu/2009/04/xkmsExt#` as defined in chapter 5.3 of [XKMSEU]²⁰.

²⁰ These extensions are subject to alignment in the context of running EU-wide "Large Scale Pilot" (lsp) projects. Concrete work on these issues is done by the project PEPPOL, see www.peppol.eu. One goal is a common

2297 **8.4.3 Verification of XKMS Validate Result Signatures**

2298 Signatures of `xkms:CompoundResult` header elements MUST be verified by nodes consuming
 2299 these header elements during the process of Content Data signatures. If signature verification fails, a
 2300 fault MUST be generated and be made available to the instance validating Content Data signatures.
 2301 Affected `xkms:CompoundResult` header element MUST NOT be consumed, certificate validation
 2302 processing MUST be reprocessed by means out of scope of this specification. It is strongly
 2303 RECOMMENDED to log this security error. Fault delivery is an implementation matter.

2304	Fault 11: SignatureOfValidateResultInvalid
2305	[Code] Sender
2306	[Subcode] SignatureOfValidateResultInvalid
2307	[Reason] Verification failed for XKMS validate result

2308 The fault [Details] property SHOULD outline the concrete verification failure.

2309 **8.5 General Processing of Custom Header Faults**

2310 Nodes a message is targeted to MUST validate the structure of the OSCI extension headers. If
 2311 syntactically invalid or not conformant to this specification, the message MUST be discarded and
 2312 following fault MUST be generated:

2313	Fault 12: MsgHeaderStructureSchemaViolation
2314	[Code] Sender
2315	[Subcode] MsgHeaderStructureSchemaViolation
2316	[Reason] On ore more OSCI header violate schema definitions

2317 More information SHOULD be given in the fault [Details] property, at least the concrete header
 2318 element the error was located in form of an XPath expression relative to the `s12:Envelope` element.

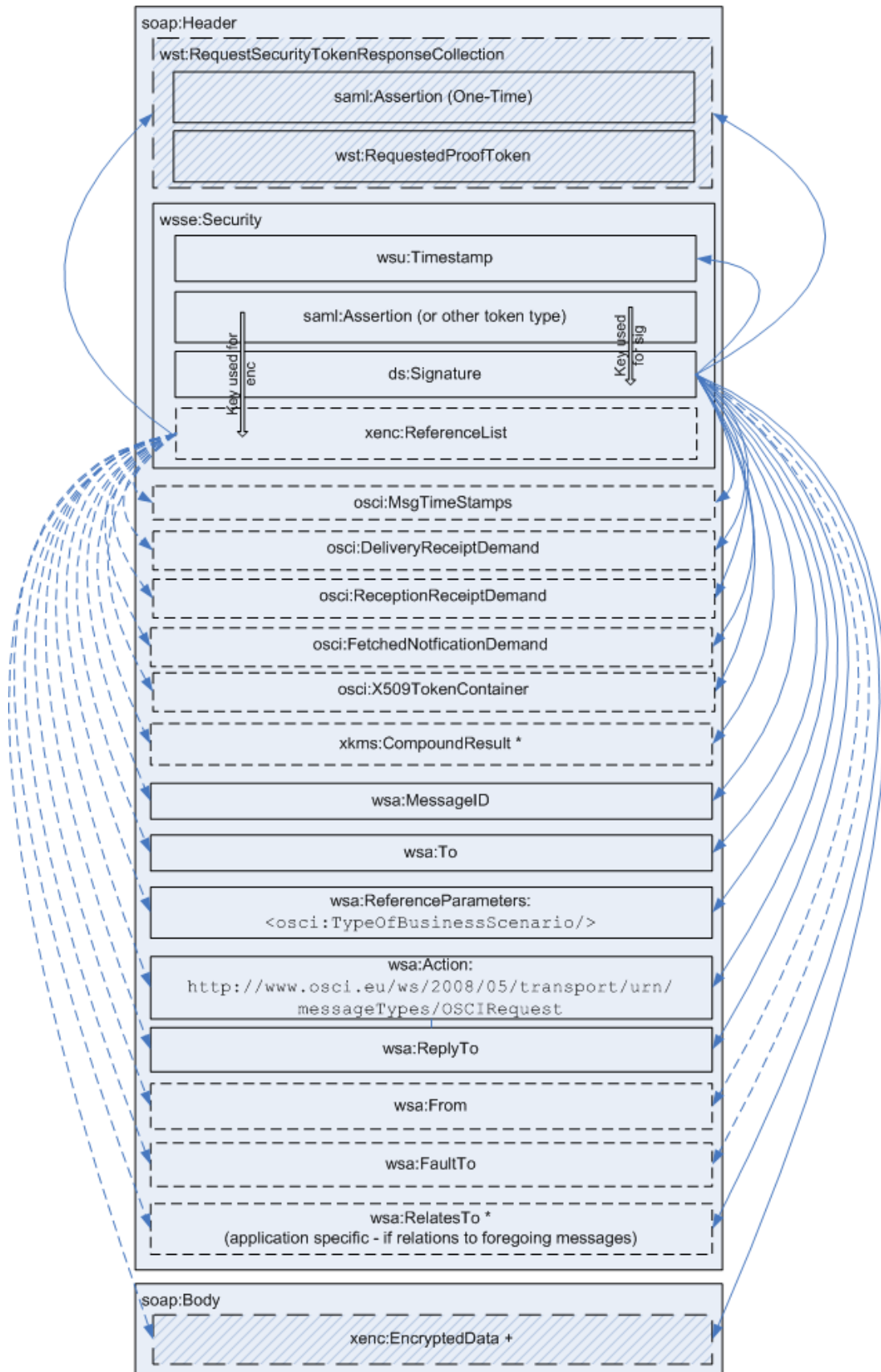
2319 **9 Constituents of OSCI Message Types**

2320 For all OSCI message types, the SOAP header and body block assemblies as well as their respective
2321 transport signature/transport encryption requirements are defined in this chapter.

2322 For a quick overview, constituents of each message type are illustrated in diagrams.

2323 In general, most header and body blocks are marked to be encrypted optionally. These blocks **MUST**
2324 be included in the transport encryption according to chapter [7], if no symmetric binding (transport over
2325 https) is used or the network between nodes involved in the message transport is secured by other
2326 precautions.

2327 **9.1 osci:Request**



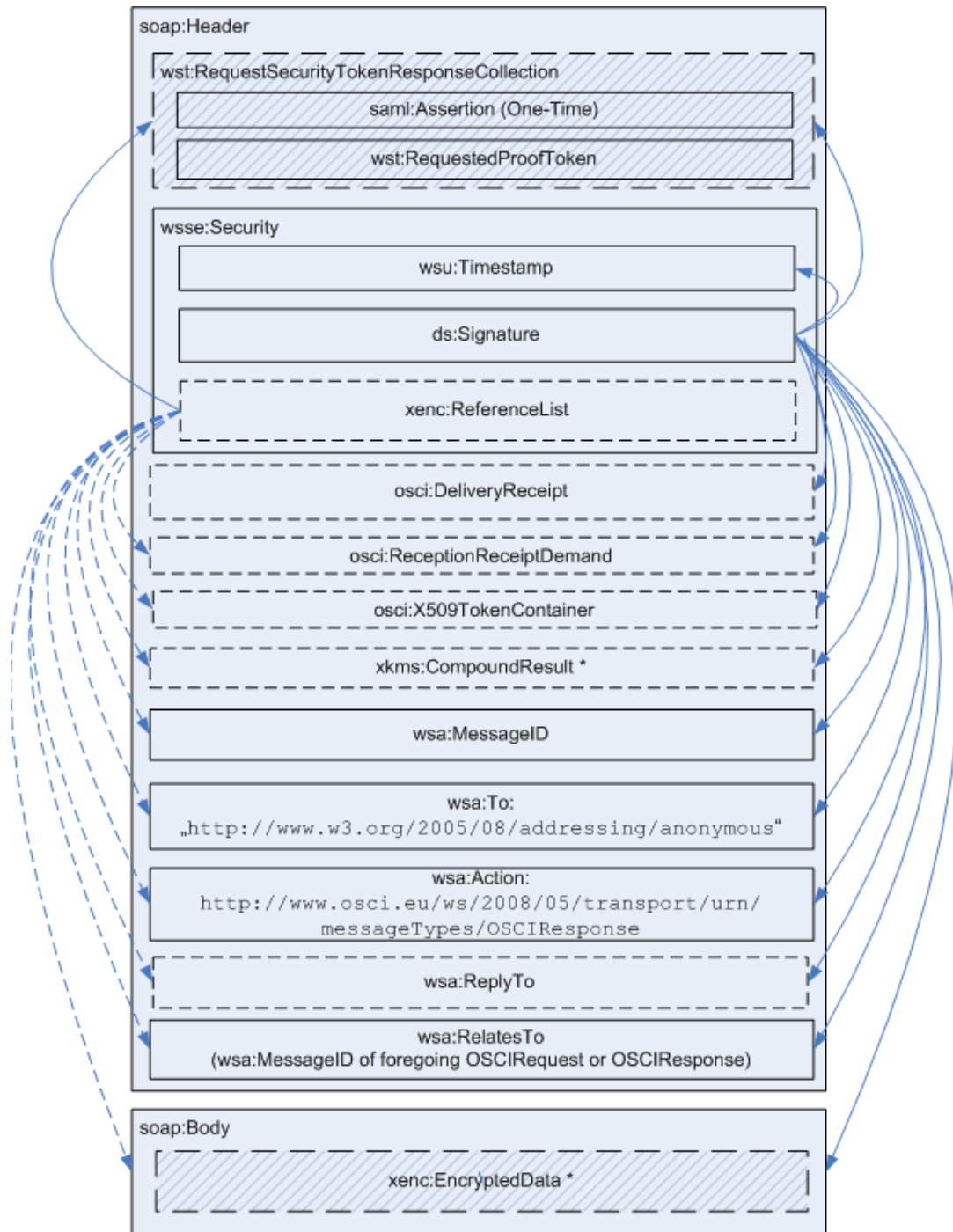
2328

2329

Figure 8: osci:Request header and body block assembly

- 2330 SOAP header blocks:
- 2331 **/wst:RequestSecurityTokenResponseCollection ?**
- 2332 This header block carries the SAML token which is needed for asynchronously delivery of
2333 receipts and notification (see chapter [7.5.5] for details). It MUST only be present, when
2334 these receipts and/or notification are required from a node in a foreign TD.
- 2335 **/wsse:Security**
- 2336 This header block MUST be present, carrying message protection data and Initiator
2337 authentication and authorization information items according the security policy of the
2338 node the message is targeted to. See chapter [7.1] for details.
- 2339 **/osci:MsgTimeStamps ?**
- 2340 This optional header block MUST only be set by the Initiator, if he wishes to supply a
2341 **.../osci:ObsoleteAfter** date in here. This header block MUST be set by a MsgBox
2342 instance and MAY be set – if not yet present - by a Recipient instance. This header block
2343 MUST always be relayed. See chapter [8.1] for details.
- 2344 **/osci:DeliveryReceiptDemand ?**
- 2345 This optional header block MUST only be set by the Initiator, if he wishes to receive a
2346 DeliveryReceipt in the backchannel response message. This header block MUST be
2347 removed from the message by the node processing it. See chapter [8.3.1.1] for details.
- 2348 **/osci:ReceptionReceiptDemand ?**
- 2349 This optional header block MUST only be set by the Initiator, if he wishes to receive a
2350 ReceptionReceipt. This header block MUST be removed from the message by the node
2351 processing it. See chapter [8.3.1.2] for details.
- 2352 **/osci:FetchedNotificationDemand ?**
- 2353 This optional header block MUST only be set by the Initiator, if he wishes to receive a
2354 FetchedNotification. This header block MUST only be processed by a MsgBox node
2355 instance and MUST be removed from the message after processing it. See chapter [8.3.3]
2356 for details.
- 2357 **/osci:X509TokenContainer ?**
- 2358 This optional header block SHOULD by provided by the Initiator, if he wishes to enable
2359 certificate validation on the message route. This header block MUST always be relayed.
2360 See chapter [8.4.1] for details.
- 2361 **/xkms:CompoundResult ?**
- 2362 This optional header block MUST by provided by nodes processing the header block
2363 **/osci:X509TokenContainer**. This header block containing
2364 **/xkms:ValidateResult** elements MUST always be relayed. See chapter [8.4.2] for
2365 details.
- 2366 **/wsa:***
- 2367 All WS-Addressing headers MUST (if in continuous blocks) /MAY (if in dashed blocks) be
2368 supplied by the Initiator and MUST always be relayed. See chapter [6.1.2] for details.
- 2369 SOAP body:
- 2370 Carries the request message ContentData, generally MUST be encrypted by the Source
2371 Application or Initiator for the Ultimate Recipient.
- 2372

2373 **9.2 osci:Response**



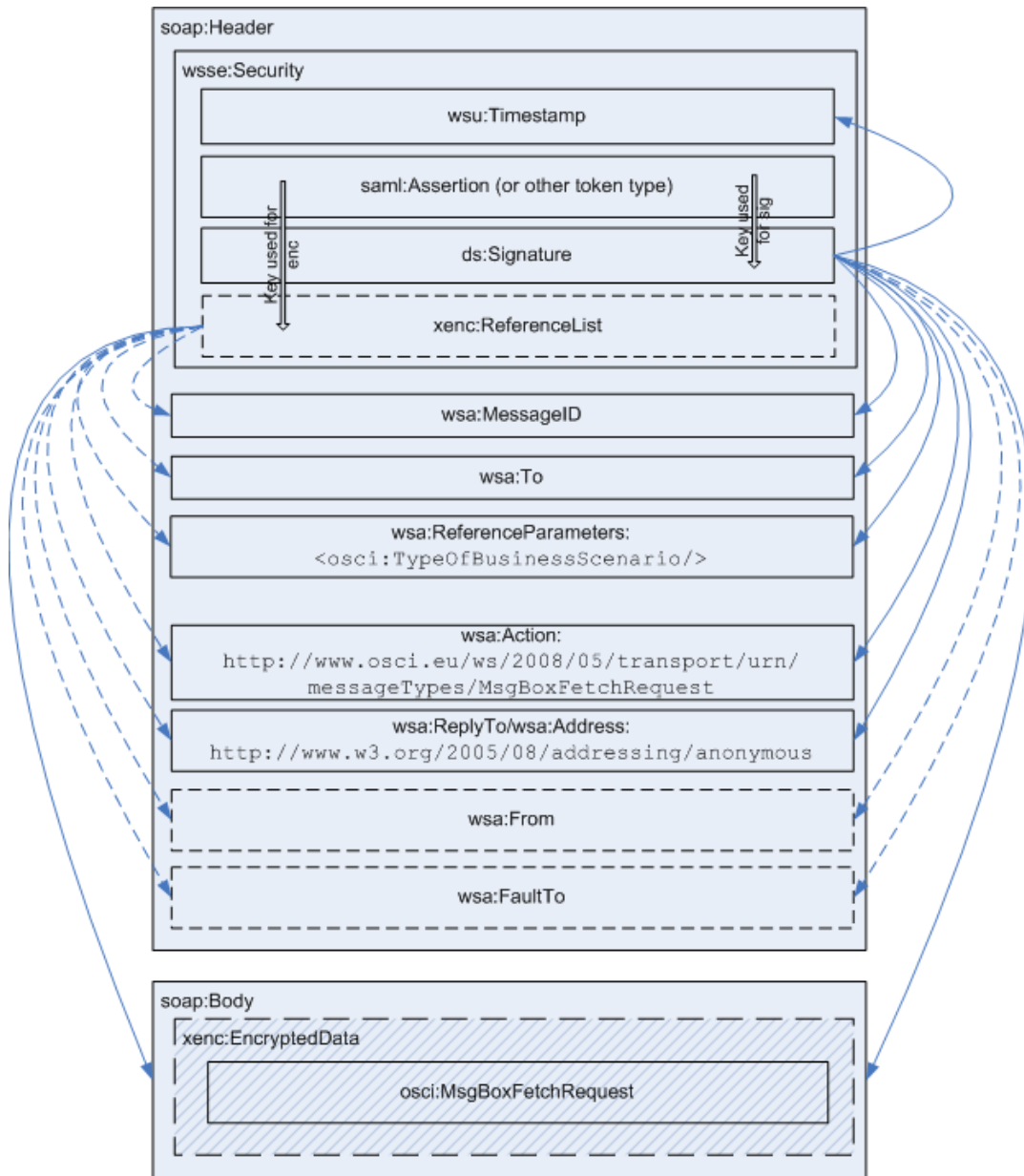
2374
2375 Figure 9: osci:Response header and body block assembly

2376 SOAP header blocks:

2377 `/wst:RequestSecurityTokenResponseCollection ?`

2378 This header block carries the SAML token which is needed for asynchronously delivery of
2379 receipts and notification (see chapter [7.5.5] for details). It MUST only be present, when
2380 these receipts and/or notification are required from a node in a foreign TD.

- 2381 **/wsse:Security**
- 2382 This header block MUST be present, carrying message protection data. See chapter [7.1]
2383 for details.
- 2384 **/osci:DeliveryReceipt ?**
- 2385 This optional header block MUST be provided by the responding endpoint, if a demand for
2386 a DeliveryReceipt is present in the corresponding request. This header block MUST not
2387 be discarded or changed on the message route. See chapter [8.3.2] for details.
- 2388 **/osci:ReceptionReceiptDemand ?**
- 2389 This optional header block MUST only be set by the responding Recipient, if he wishes to
2390 receive a ReceptionReceipt for the response message. This header block MUST NOT be
2391 set by a MsgBox instance. This header block MUST be removed from the message by the
2392 node processing it. See chapter [8.3.1.2] for details.
- 2393 **/osci:X509TokenContainer ?**
- 2394 This optional header block SHOULD be provided by the responding Recipient, if he
2395 wishes to enable certificate validation on the message route. This header block SHOULD
2396 NOT be set by a MsgBox instance. This header block MUST always be relayed. See
2397 chapter [8.4.1] for details.
- 2398 **/xkms:CompoundResult ?**
- 2399 This optional header block MUST be provided by nodes processing the header block
2400 **/osci:X509TokenContainer**. This header block containing
2401 **/xkms:ValidateResult** elements MUST always be relayed. See chapter [8.4.2] for
2402 details.
- 2403 **/wsa:***
- 2404 All WS-Addressing headers MUST (if in continuous blocks) /MAY (if in dashed blocks) be
2405 supplied by the resending endpoint and MUST always be relayed. See chapter [6.1.2] for
2406 details.
- 2407 SOAP body:
- 2408 May carry the response message ContentData in case of point-to-point scenarios. If
2409 present, it generally MUST be encrypted by the Target Application or Recipient for the
2410 Initiator. If an error occurred, a fault message is placed here instead.

2411 **9.3 MsgBoxFetchRequest**

2412

2413

Figure 10: MsgBoxFetchRequest header and body block assembly

2414 SOAP header blocks:

2415 **/wsse:Security**

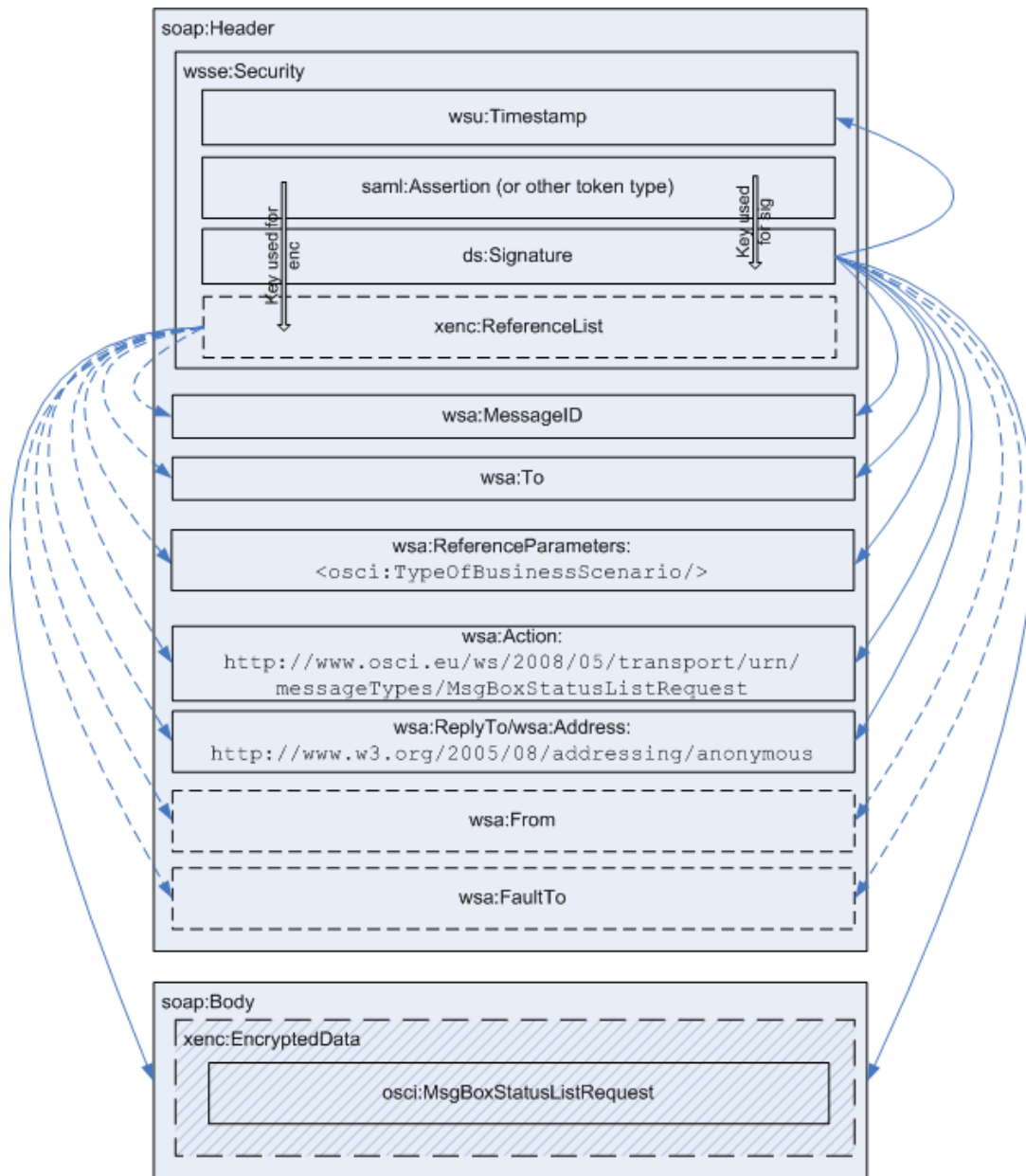
2416 This header block **MUST** be present, carrying message protection data and requestor
 2417 (MsgBox owner in this case) authentication and authorization information items according
 2418 the security policy MsgBox instance. See chapter [7.1] for details.

2419 **/wsa:***

2420 All WS-Addressing headers **MUST** (if in continuous blocks) /**MAY** (if in dashed blocks) be
 2421 supplied by the Initiator. See chapter [6.1.2] and [8.2.1] for details.

2422 SOAP body:

2423 Carries the details of the `MsgBoxFetchRequest`, generally **MUST** be transport encrypted.
 2424 See chapter [8.2.1] for details.

2425 **9.4 MsgBoxStatusListRequest**

2426

2427

Figure 11: MsgBoxStatusListRequest header and body block assembly

2428 SOAP header blocks:

2429 **/wsse:Security**

2430 This header block MUST be present, carrying message protection data and requestor
 2431 (MsgBox owner in this case) authentication and authorization information items according
 2432 the security policy MsgBox instance. See chapter [7.1] for details.

2433 **/wsa:***

2434 All WS-Addressing headers MUST (if in continuous blocks) /MAY (if in dashed blocks) be
 2435 supplied by the Initiator. See chapter [6.1.2] and [8.2.2] for details.

2436 SOAP body:

2437 Carries the details of the MsgBoxFetchRequest, generally MUST be transport encrypted.
 2438 See chapter [8.2.2] for details.

2439 **9.5 MsgBoxResponse**

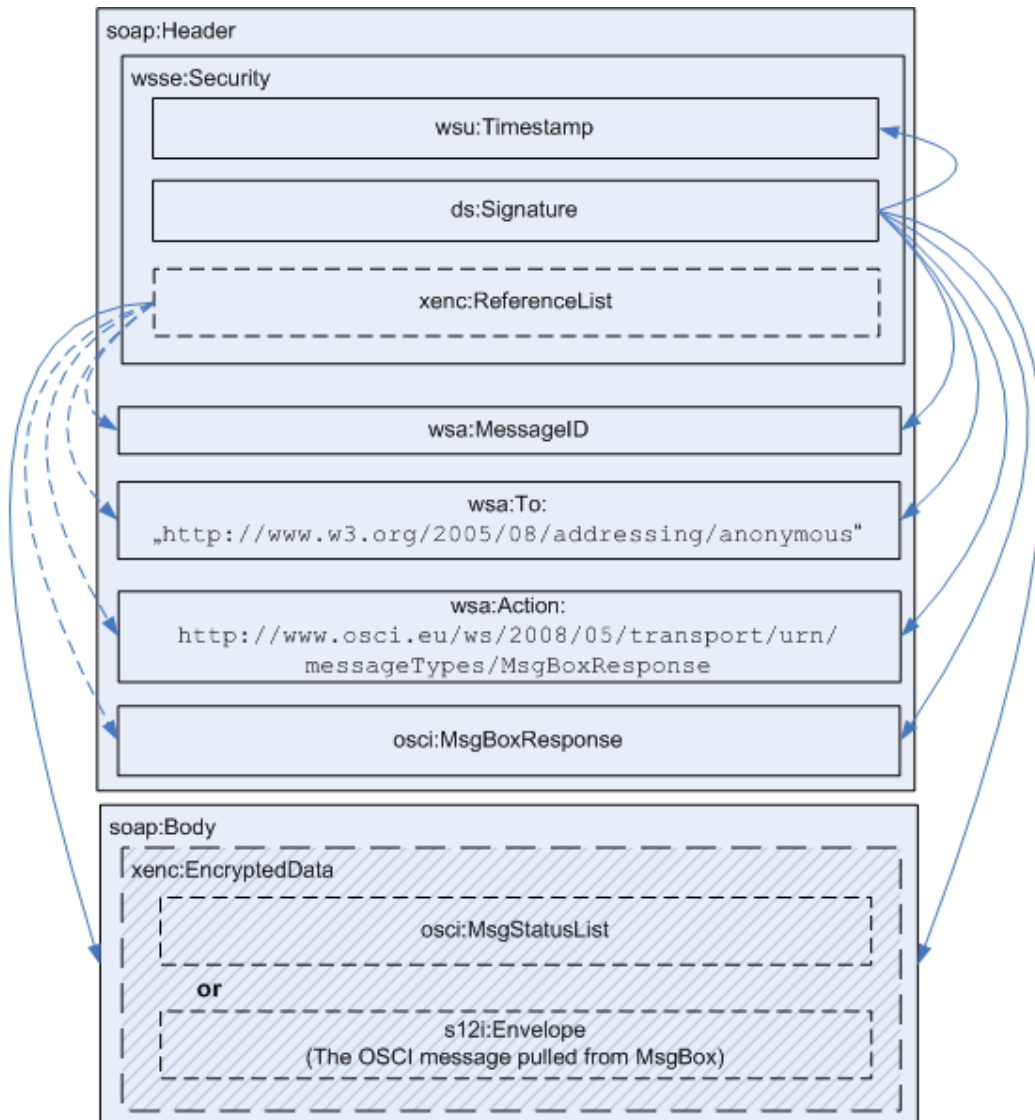


Figure 12: MsgBoxResponse header and body block assembly

2440

2441

2442 SOAP header blocks:

2443 **/wsse:Security**

2444 This header block MUST be present, carrying message protection data. See chapter [7.1]
2445 for details.

2446 **/wsa:***

2447 All WS-Addressing headers MUST (if in continuous blocks) /MAY (if in dashed blocks) be
2448 supplied by the Initiator. See chapter [6.1.2] and [8.2.3] for details.

2449 **/osci:MsgBoxResponse**

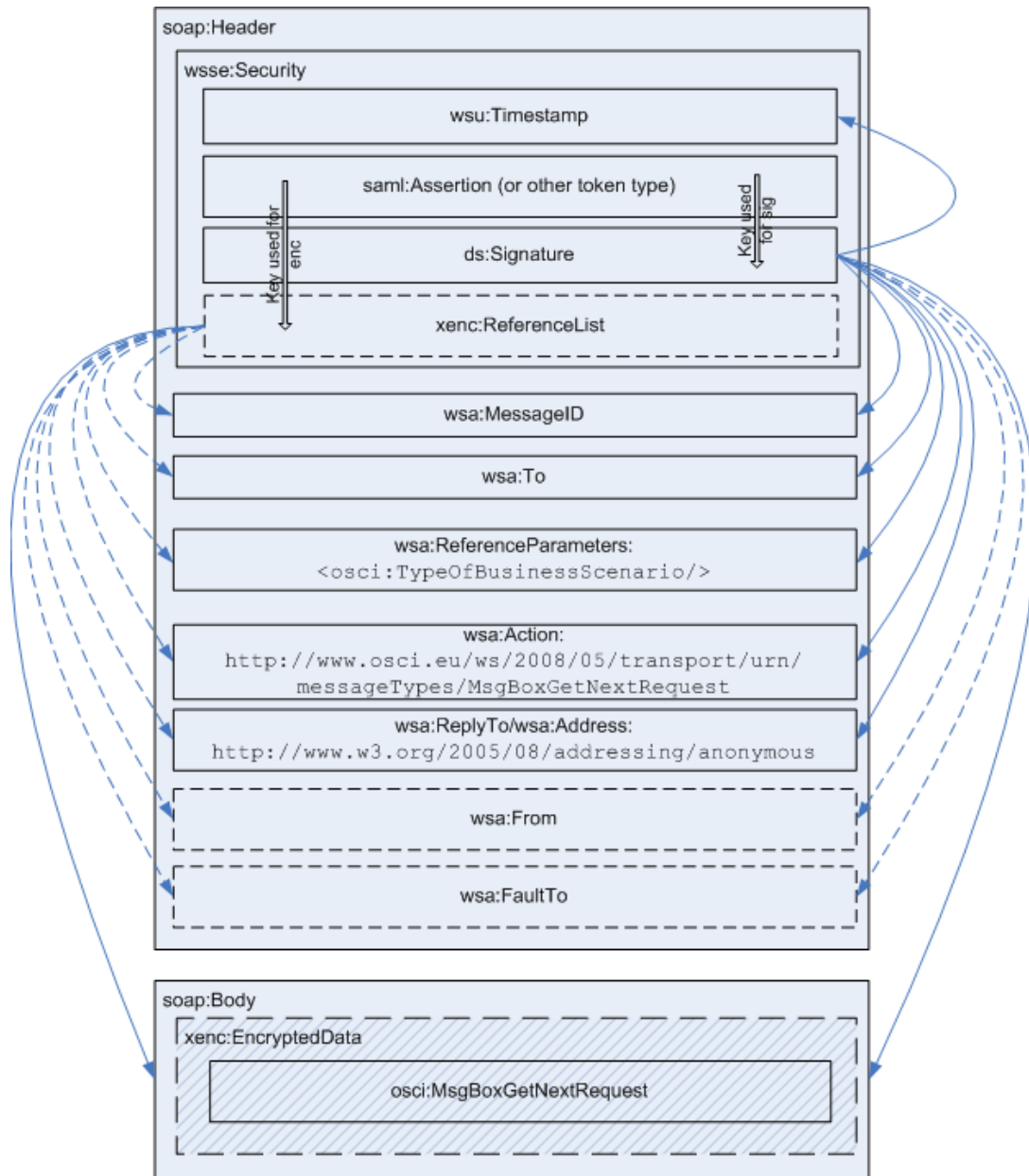
2450 This header carrying status information concerning the actual message box access MUST
2451 be set by the resending MsgBox instance. See chapter [8.2.3] for details.

2452 SOAP body:

2453 Carries the requested message status list or the message fetched from the MsgBox –
2454 depending on the initial request. It generally MUST be transport encrypted. See chapter

2455 [8.2.3.1] and [8.2.3.2] for details. If an error occurred, a fault message is placed here
 2456 instead.

2457 **9.6 MsgBoxGetNextRequest**



2458
 2459 Figure 13: MsgBoxGetNextRequest header and body block assembly

2460 SOAP header blocks:

2461 **/wsse:Security**

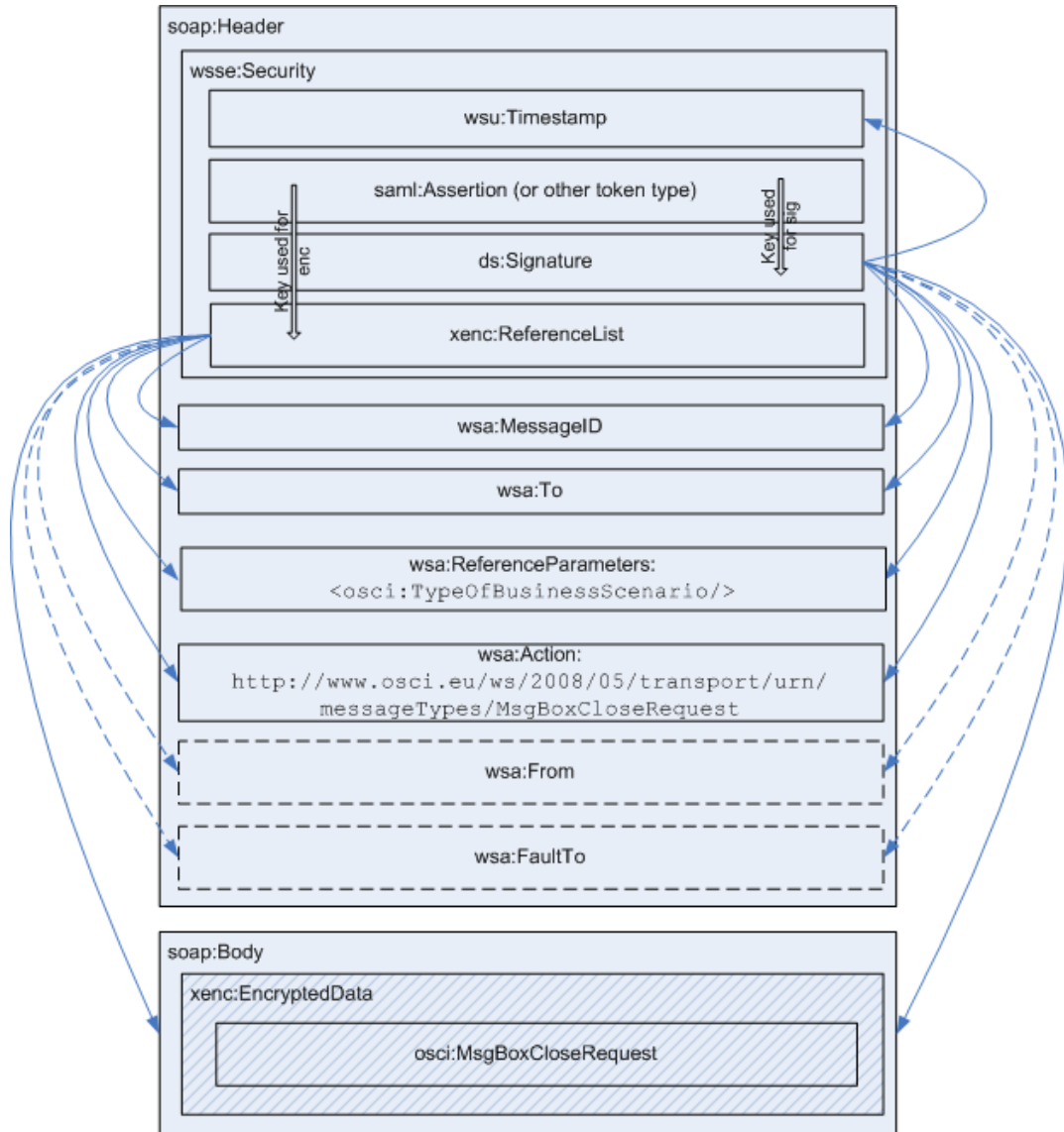
2462 This header block MUST be present, carrying message protection data and requestor
 2463 (MsgBox owner in this case) authentication and authorization information items according
 2464 the security policy MsgBox instance. See chapter [7.1] for details.

2465 **/wsa:***

2466 All WS-Addressing headers MUST (if in continuous blocks) /MAY (if in dashed blocks) be
 2467 supplied by the Initiator. See chapter [6.1.2] and [8.2.4] for details.

- 2468 SOAP body:
- 2469 Carries the details of the MsgBoxGetNextRequest, generally MUST be transport
- 2470 encrypted. See chapter [8.2.4] for details.

2471 **9.7 MsgBoxCloseRequest**



2472
2473 Figure 14: MsgBoxClose header and body block assembly

2474 SOAP header blocks:

2475 **/wsse:Security**

2476 This header block MUST be present, carrying message protection data and requestor
2477 (MsgBox owner in this case) authentication and authorization information items according
2478 the security policy MsgBox instance. See chapter [7.1] for details.

2479 **/wsa:***

2480 All WS-Addressing headers MUST (if in continuous blocks) /MAY (if in dashed blocks) be
2481 supplied by the Initiator. See chapter [6.1.2] and [8.2.5] for details.

-
- 2482 SOAP body:
- 2483 Carries the details of the MsgBoxCloseRequest, generally MUST be transport encrypted.
- 2484 See chapter [8.2.5] for details.

2485 **10 Policies and Metadata of Communication Nodes and** 2486 **Endpoints**

2487 **10.1 General usage of Web Service Description Language**

2488 The Web Service Description Language (WSDL) provides a broadly-adopted foundation on which
2489 interoperable Web Services can be build. WS-Policy Framework [WSPF] and WS-Policy Attachment
2490 [WSPA] collectively define a framework, model and grammar for expressing the requirements, and
2491 general characteristics of entities in an XML Web Services-based system.

2492 In general, endpoint properties and requirements MUST be described in machine readable form of
2493 WSDLs and policies. For sake of interoperability with currently available implementations of the WS-
2494 Stack, this specification restricts to Web Service Description Language 1.1 [WSDL11].

2495 This specification does not assume a mandatory mechanism how WSDLs of endpoints must be made
2496 available. To facilitate retrieval and online exchange of WSDLs, they SHOULD be exposed in
2497 appropriate directories OSCI communication networks; the base established mechanism to access a
2498 WSDL of a concrete Web Service endpoint is a http(s) GET-Request in the form

2499 `http(s)://endpoint-url?WSDL.`

2500 Conformant implementations SHOULD at least support this mechanism. The specification WS
2501 Metadata Exchange [WSMEX] describes a more sophisticated way to encapsulate services metadata
2502 and a protocol to retrieve it. It allows the client to interact with the service automatically, fetch all
2503 relevant metadata and aids the client in self-configuring. Support of WS Metadata Exchange is
2504 strongly RECOMMENDED.

2505 **NOTE on WSDL/Policy integrity:** Policies and WSDL-Files MUST be secured by digital signatures to
2506 allow detection of possible corruptions. Unfortunately, there is no standard format and placement
2507 defined so far by the WS-Policy Framework for adequate digital signatures, what obviously results in
2508 the lack of integrity check mechanisms in known framework implementations when accessing policies
2509 and WSDL-Files. An appropriate specification and recommendation for implementors will be published
2510 by the OSCI Steering Office mid 2009 after finishing actually running tests on solution variants
2511 addressing this issue.²¹

2512 Endpoints are not forced to expose their properties and requirements in form of online available and
2513 machine readable WSDLs and/or policies. Developers may exchange this information on informal
2514 basis out of scope of this specification (i.e. word-of-mouth, documentation).

2515 **NOTE on WSDL/Policy examples:** Patterns of WSDL instances and reference policies for classes of
2516 OSCI based scenarios will be developed in a distinct project and be made available in step by step in
2517 2009 as addendum to this document.

2518 **Technical NOTE for policy instances:** All policies defined for an endpoint MUST carry an Id-
2519 Attribute for the outer element `/wsp:Policy/@wsu:Id` to be referenceable for policy attachment
2520 and metadata exchange purposes.

2521 **10.1.1 WSDL and Policies for MEP synchronous point-to-point**

2522 For this communication scenario, description of endpoint requirements and abilities SHOULD be
2523 outlined in one WSDL containing all services and ports with their respective policies available here.
2524 Following general requirements MUST be considered when designing WSDL instances:

²¹ This work is done in the context of the OSCI Profiling project and will be published as an addendum to this specification. Results are planned to be brought to the appropriate OASIS standardization body.

2525 **R1200** - A `/wsdl111:port` MUST always contain an entry `/wsa:EndpointReference` with the
 2526 `.../wsa:Address` element as well as `.../wsa:ReferenceParameters` outlining the URI of
 2527 the `.../osci:TypeOfBusinessScenario` served by this port²². Each specific
 2528 `/osci:TypeOfBusinessScenario` itself correlates to a concrete Content Data
 2529 message structure given by the `/wsdl111:port` reference chain to a `/wsdl111:binding`
 2530 and `/wsdl111:portType` entry in this WSDL instance.

2531 10.1.2 WSDL and Policies for asynchronous MEPs via Message Boxes

2532 These MEPs at least have two endpoints in view a message is targeted to:

- 2533 • Initially, a Source Application has to build up the SOAP body content according to a concrete
 2534 schema bound to the actual underlying `/osci:TypeOfBusinessScenario`. In addition,
 2535 security requirements bound to the Recipient apply like End-to-end encryption and digital
 2536 signatures to be applied to Content Data, which SHOULD be expressed by according WS
 2537 Security Policy expressions.
- 2538 • For transport to the Recipients `MsgBox` instance, the WSDL and policies of this target node
 2539 apply. For every `/osci:TypeOfBusinessScenario` accepted here, the body structure is of
 2540 type `xenc:EncryptedData`. The WS Security Policy if effect here MUST NOT lead to initiate
 2541 body decryption processing, as the therefore needed private encryption key is only known to
 2542 the Recipient node.

2543 As of today known WS-Framework implementations, WS Security Policies attached in the WSDL of
 2544 the node a message is targeted to are completely in effect at the targeted node; it is not possible to
 2545 bind them e.g. to a specific `s12:role` without in-depth change of processing logic of standard WS-
 2546 Framework implementations.

2547 To solve this problem, for this version of the OSCI Transport specification following recommendation
 2548 applies²³:

- 2549 • The `MsgBox` node exposes the WSDL and policies according his needs on opaque body,
 2550 transport security and authentication/authorization per accepted
 2551 `/osci:TypeOfBusinessScenario`.
- 2552 • WSDL and policies in effect for the Recipient node are referenced or contained in the
 2553 `/wsa:EndpointReference/wsa:Metadata` element as described in chapter [6.1.1], bound
 2554 to the `/wsdl111:port` policy attachment point.

2555 10.2 OSCI specific Characteristics of Endpoints

2556 To enable OSCI endpoints to describe their requirements and capabilities, this specification defines
 2557 OSCI policy assertions that leverage the WS-Policy framework. In general, it is RECOMMENDED to
 2558 attach the policy assertions defined here to a to a port [WSDL11] respective endpoint [WSDL20] policy
 2559 subject.

2560 10.2.1 Certificates used for Signatures and Encryption

2561 For OSCI based message exchange, X.509v3-Certificates MUST be used for following purposes:

- 2562 • Encryption to be processed on Initiator side
 - 2563 ○ End-to-end encryption of Content Data targeted from a Source Application to a
 2564 Target Application

²² following chapter [6.1.1], use of WS-Addressing in OSCI

²³ A WSDL/Policies template for this MEP as well as `MsgBox` access thru the recipient will be made available as addendum immediately after publishing this specification

- 2565 ○ Transport encryption, in cases where asymmetric encryption is required by a
2566 specifics application scenario – in general expressed by an adequate security policy
- 2567 ● Certificates used for signatures at Recipient side (respective his MsgBox service); an Initiator
2568 MAY – in cases of doubt - cross-check whether received signatures are generated with the
2569 certificates exposed in this endpoint policy (detection of possible man-in-the-middle attacks):
- 2570 ○ Signature application for OSCI receipts and possible other message parts – in cases
2571 where the signature must be useable for long term provableness
- 2572 ○ If offered: Generation of cryptographic time stamps.

2573 Additional application purposes MAY be defined and supported by dedicated implementations.

2574 Syntax for the OSCI policy containing assertions for X.509v3-Certificates usages:

```

2575 <wsp:Policy wsu:Id="xs:ID">
2576   <osci:X509CertificateAssertion>
2577     <wsp:ALL>
2578       <wsse:SecurityTokenReference wsu:Id="xs:ID" ?
2579         Usage=
2580         "http://www.osci.eu/2008/05/common/names/TokenUsage/e2eContentEncryption"
2581       |
2582         "http://www.osci.eu/2008/05/common/names/TokenUsage/TransportEncryption"
2583       |
2584         "http://www.osci.eu/2008/05/common/names/TokenUsage/ReceiptSigning"
2585       |
2586         "http://www.osci.eu/2008/05/common/names/TokenUsage/TSPSigning" *
2587       osci:Role=
2588         "http://www.osci.eu/ws/2008/05/transport/role/Recipient" |
2589         "http://www.osci.eu/ws/2008/05/transport/role/MsgBox" |
2590         "http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver" * >
2591
2592       ( <wsse:Embedded ValueType="xs:anyURI" ? >
2593         <wsse:BinarySecurityToken wsu:Id="xs:ID" ?
2594           ValueType=
2595             "http://docs.oasis-open.org/wss/2004/01/
2596               oasis-200401-wss-x509-token-profile-1.0#X509v3"
2597           EncodingType=
2598             "http://docs.oasis-open.org/wss/2004/01/
2599               oasis-200401-wss-soapmessage-security-1.0#Base64Binary" >
2600             xs:base64Binary
2601           </wsse:BinarySecurityToken>
2602         </wsse:Embedded> )
2603       |
2604       ( <wsse:Reference URI="xs:anyUri"
2605         ValueType=
2606           "http://docs.oasis-open.org/wss/2004/01/
2607             oasis-200401-wss-x509-token-profile-1.0#X509v3" /> )
2608       |
2609       ( <wsse:KeyIdentifier wsu:Id="xs:ID" ?
2610         ValueType=
2611           "http://docs.oasis-open.org/wss/
2612             oasis-wss-soap-message-security-1.1#ThumbprintSHA1"
2613         EncodingType=
2614           "http://docs.oasis-open.org/wss/2004/01/
2615             oasis-200401-wss-soapmessage-security-1.0#Base64Binary">
2616           xs:base64Binary
2617         </wsse:KeyIdentifier> )
2618
2619     </wsse:SecurityTokenReference
2620   </wsp:ALL>
2621 </osci:X509CertificateAssertion>
2622 </wsp:Policy>

```

2623 Description of elements and attributes in the schema overview above:

2624 **/wsp:Policy**

2625 The whole assertion MUST be embedded in a policy block according to WS Policy.

- 2626 **/wsp:Policy/@wsu:Id**
- 2627 To be referenceable, the policy MUST carry an attribute of type **xs:ID**.
- 2628 **/wsp:Policy/osci:X509CertificateAssertion**
- 2629 The policy block containing all assertions.
- 2630 **/wsp:Policy/osci:X509CertificateAssertion/wsp:ALL**
- 2631 Following the semantics of WS Policy Framework [WSPF], all behaviours represented by
- 2632 the assertions embedded in this block are required/valid.
- 2633 **/wsp:Policy/osci:X509CertificateAssertion/wsp:ALL/wsse:SecurityTokenReferen**
- 2634 **ce**
- 2635 This element defined in WS Security [WSS] MUST be used as container for a single
- 2636 X.509v3-Certificate (or a reference to it) and its attributes.
- 2637 As all single certificate details are contained in this block, for brevity full path qualification
- 2638 **/wsp:Policy/osci:X509CertificateAssertion/wsp:ALL/wsse:SecurityTokenReferen**
- 2639 **ce** is symbolized by [SingleToken] in the following descriptions.
- 2640 **[SingleToken]/@wsu:id**
- 2641 A certificate contained/described in this policy MUST be uniquely referenceable – i.e.,
- 2642 from other policies describing the same endpoint. This attribute of type **xs:ID** MUST
- 2643 carry an appropriate unique value.
- 2644 **[SingleToken]/@Usage**
- 2645 This attribute defines the purposes a certificate is used for, at least one of the URIs
- 2646 outlined above MUST be supplied as value in this attribute of type list of **xs:anyURI**.
- 2647 Predefined usage semantics are:

Usage for	URI
End-to-end encryption of Content Data	http://www.osci.eu/2008/05/common/names/TokenUsage/e2eContentEncryption
Asymmetric transport encryption	http://www.osci.eu/2008/05/common/names/TokenUsage/TransportEncryption
Signature of OSCI receipts; also applicable for signatures of other message parts	http://www.osci.eu/2008/05/common/names/TokenUsage/ReceiptSigning
Generation of cryptographic time stamps	http://www.osci.eu/2008/05/common/names/TokenUsage/TSPSigning

2648 Table 8: OSCI X.509-Token usages

- 2649 **[SingleToken]/@osci:Role**
- 2650 This attribute defines logical roles a certificate is assigned to, at one of the URIs outlined
- 2651 above MUST be supplied value in this attribute of type list of **xs:anyURI**. Regularly, a
- 2652 single certificate SHOULD NOT be assigned to more the one role; as constellations are
- 2653 imaginable, where logical roles are pooled – like for a Recipient and Ultimate Recipient
- 2654 which are using the same signature certificate - is these cases more then one role
- 2655 assignment MAY be used.
- 2656 For example, this role attribute allows Initiators to control, whether a receipt is signed with
- 2657 the right certificate used by the septic receipt issuer role outlined in the receipt.
- 2658 Predefined logical roles are:

Usage for	URI
OSCI Recipient – i.e. using this certificate for signing DeliveryReceipts in synchronous case or as transport encryption certificate	http://www.osci.eu/ws/2008/05/transport/role/Recipient
Ultimate receiver in the sense of [SOAP12]; i.e. an Ultimate Recipient using this certificate for end-to-end encryption or ReceptionReceipt signing	http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver
MsgBox service node; i.e. may have his own transport encryption certificate (as MUST be used for "one time tokens")	http://www.osci.eu/ws/2008/05/common/names/role/MsgBox

2659 Table 9: SOAP/OSCI roles assigned to token usages

2660 **R1200** - Inside a [SingleToken], the certificate itself may be embedded, referenced or identified by
 2661 a thumbprint. Other choices foreseen by WS-Security for
 2662 `/wsse:SecurityTokenReference` MUST NOT be used.

2663 Choice for embedded tokens

2664 `[SingleToken]/wsse:Embedded`

2665 This choice MUST be taken for embedding X.509v3-Certificates. It is strongly
 2666 RECOMMENDED to use this choice for certificates to be used for encryption purposes, as
 2667 an Initiator and STS may need the respective public key for encryption. Referencing those
 2668 certificates could cause additional to network connection needs.

2669 `[SingleToken]/wsse:Embedded/@ValueType`

2670 This element MAY carry this attribute of type `xs:anyURI`. It is not used in the context of this
 2671 policy.

2672 `[SingleToken]/wsse:Embedded/wsse:BinarySecurityToken`

2673 Generic container to carry security tokens in binary format; MUST contain the X.509v3-
 2674 Certificate in base64Binary format.

2675 `[SingleToken]/wsse:Embedded/wsse:BinarySecurityToken/@wsu:Id`

2676 This attribute of type `xs:ID` is optional. It is not used in the context of this policy.

2677 `[SingleToken]/wsse:Embedded/wsse:BinarySecurityToken/@ValueType`

2678 As only X.509v3-Certificates are described/contained here, the URI outlined above MUST
 2679 be supplied as value in this attribute of type of `xs:anyURI`.

2680 `[SingleToken]/wsse:Embedded/wsse:BinarySecurityToken/@EncodingType`

2681 Hence X.509v3-Certificates MUST be encoded in base64Binary format here, the URI
 2682 outlined above MUST be supplied as value in this attribute of type of `xs:anyURI`.

2683 Choice for directly referencing tokens

2684 `[SingleToken]/wsse:Reference`

2685 This choice MUST be taken if referencing X.509v3-Certificates stored otherwise.

2686 `[SingleToken]/wsse:Reference/@URI`

2687 This attribute of type `xs:anyURI` MUST identify a X.509v3-Certificate. If a fragment is
 2688 specified, then it indicates the local ID of the security token being referenced. The URI
 2689 MUST NOT identify a `/wsse:SecurityTokenReference` element, a
 2690 `/wsse:Embedded` element, a `/wsse:Reference` element, or a
 2691 `/wsse:KeyIdentifier` element.

2692 `[SingleToken]/wsse:Reference/@ValueType`

2693 As only X.509v3-Certificates are described/contained here, the URI outlined above MUST
 2694 be supplied as value in this attribute of type of `xs:anyURI`.

2695 Choice for referencing tokens by thumbprint

2696 This choice SHOULD NOT be used for certificates to be used for encryption purposes, as this may
 2697 burden Initiator and STS to locate the needed public key for encryption.

2698 `[SingleToken]/wsse:KeyIdentifier`

2699 **R1210:** This choice MUST be taken if referencing X.509v3-Certificates by thumbprint.
 2700 Other choices foreseen by WS-Security for `/wsse:KeyIdentifier` MUST NOT be
 2701 used.

2702 `[SingleToken]/wsse:KeyIdentifier/@wsu:Id`

2703 This attribute of type `xs:ID` is optional. It is not used in the context of this policy.

2704 `[SingleToken]/wsse:KeyIdentifier/@ValueType`

2705 As only thumbprints are allowed for referencing here, the URI outlined above MUST be
 2706 supplied as value in this attribute of type of `xs:anyURI`.

2707 `[SingleToken]/wsse:KeyIdentifier/@EncodingType`

2708 Hence thumbprints MUST be encoded in base64Binary format here, the URI outlined
 2709 above MUST be supplied as value in this attribute of type of `xs:anyURI`.

2710 **NOTE on usage of alternate certificates for the same purpose and role:**

2711 If more than one certificate may be used for a combination of `[SingleToken]/@osci:Role` and
 2712 `[SingleToken]/@wsse:Usage`, these policy elements `/wsse:SecurityTokenReference`
 2713 MUST be grouped in a `/wsp:ExactlyOne` container to express that only one of the alternatives may
 2714 be chosen.

2715 10.2.2 Endpoint Services and Limitations

2716 OSCI Recipients respective MsgBox services MAY offer/expose following services and limits:

- 2717 • Qualified timestamp application for signatures; this service is requestable by an Initiator for
 2718 receipts;
- 2719 • Message lifetime control; this service interprets the
 2720 `/osci:MsgTimeStamps/osci:ObsoleteAfter` SOAP header element probably set by an
 2721 Initiator. This marker only makes sense in asynchronous MEPs, hence the processing policy
 2722 assigned to is only of interest for MsgBox instances;
- 2723 • Maximum accepted message size and acceptance frequency per hour.

2724 Syntax for OSCI endpoint services policy assertions:

```
2725 <wsp:Policy wsu:Id="xs:ID">
2726
2727   <osci:QualTSPAssertion PolicyRef="xs:anyURI" ? > ?
2728
2729   <osci:ObsoleteAfterAssertion PolicyRef="xs:anyURI" ? > ?
2730     <osci:MsgRetainDays>
2731       xs:positiveInteger
2732     </osci:MsgRetainDays> ?
```

```

2733 <osci:WarningMsgBeforeObsolete>
2734   xs:nonNegativeInteger
2735 </osci:WarningMsgBeforeObsolete> ?
2736 </osci:ObsoleteAfterAssertion> ?
2737
2738 <osci:MsgLimitsAssertion>
2739   <osci:MaxSize>
2740     xs:positiveInteger
2741   </osci:MaxSize> ?
2742   <osci:MaxPerHour>
2743     xs:positiveInteger
2744   </osci:MaxPerHour> ?
2745 </osci:MsgLimitsAssertion> ?
2746
2747 <wsp:Policy>

```

2748 Description of elements and attributes in the schema overview above:

2749 **/wsp:Policy**

2750 The whole assertion **MUST** be embedded in a policy block according to WS Policy.

2751 **/wsp:Policy/@wsu:Id**

2752 To be referenceable, the policy **MUST** carry an attribute of type **xs:ID**.

2753 **/wsp:Policy/osci:QualTSPAssertion ?**

2754 The presence of this element signals the availability of a qualified timestamp service.

2755 **/wsp:Policy/osci:QualTSPAssertion/@PolicyRef ?**

2756 This optional attribute of type **xs:anyURI** **SHOULD** be provided and carry a link to i.e. human readable policies describing terms and conditions under which this service is made available.

2759 **/wsp:Policy/osci:ObsoleteAfterAssertion ?**

2760 The presence of this element signals the fact this endpoint will care about a SOAP header entry **/osci:MsgTimeStamps/osci:ObsoleteAfter**.

2762 **/wsp:Policy/osci:ObsoleteAfterAssertion/@PolicyRef ?**

2763 This optional attribute of type **xs:anyURI** **MAY** be provided and carry a link to i.e. human readable policies describing terms and conditions about deletion of messages marked to be obsolete meanwhile.

2766 **/wsp:Policy/osci:ObsoleteAfterAssertion/MsgRetainDays ?**

2767 This optional element of type **xs:positiveInteger** **SHOULD** be provided to expose the number of days a message still is hold available after the date provided by the **.../osci:ObsoleteAfter** entry.

2770 **/wsp:Policy/osci:ObsoleteAfterAssertion/WarningBeforeObsolete ?**

2771 This optional element of type **xs:nonNegativeInteger** **SHOULD** be provided to expose the number of days a warning is generated before the date provided by the **.../osci:ObsoleteAfter** entry. Thus, an escalation procedure could be triggered for messages seen to be of high importance. How this warning is generated and delivered is a matter of implementation of this service and **SHOULD** be described in the terms and conditions policy.²⁴

2777 **/wsp:Policy/osci:MsgLimitsAssertion ?**

²⁴ This warning could i.e. be delivered in the body of an `osci:Request` to the Initiator alike the `FetchNotification` message.

2778 The presence of this element signals the fact this endpoint has restrictions for incoming
2779 messages .

2780 **/wsp:Policy/osci:MsgLimitsAssertion/MaxSize ?**

2781 This optional element of type **xs:positiveInteger** outlines the maximum size in
2782 kilobytes for incoming messages this endpoint accepts.

2783 If an incoming message exceeds this limit, it **MUST** be withdrawn and a fault **MUST** be
2784 returned to the targeting node:

2785	Fault 13: MsgSizeLimitExceeded
2786	[Code] Sender
2787	[Subcode] MsgSizeLimitExceeded
2788	[Reason] Message size exceeds policy

2789 **/wsp:Policy/osci:MsgLimitsAssertion/MaxPerHour ?**

2790 This optional element of type **xs:positiveInteger** outline the maximum amount in of
2791 messages accepted per hour from the same originating node²⁵ .

2792 If an incoming messages originated from the same targeting node exceed this limit, the
2793 message **MUST** be withdrawn and a fault **MUST** be returned to the targeting node:

2794	Fault 14: MsgFrequencyLimitExceeded
2795	[Code] Sender
2796	[Subcode] MsgFrequencyLimitExceeded
2797	[Reason] Message frequency per hour exceeds policy

2798 **10.3 WS Addressing Metadata and WS MakeConnection**

2799 Hence the use of WS-Addressing is mandatory for OSCI, an endpoint policy **MUST** contain WS-
2800 Addressing properties described here in terms of WS-Addressing Metadata [WSAM] .

2801 Following policy assertions **MUST** be bound to the **ws1d11:port** (WSDL 2.0: endpoints) or
2802 **wsd111:binding** endpoint policy subjects which accept messages of type **osci:Request**; WS
2803 MakeConnection is not supported in this case:

```
2804 <wsp:Policy wsu:Id="xs:ID" ?>
2805   <wsam:Addressing>
2806     <wsp:Policy/> ?
2807   </wsam:Addressing>
2808 </wsp:Policy>
```

2809 This policy ascertains the use of WS-Addressing and that the endpoint requires request messages to
2810 use response endpoint EPRs that contain something other than the anonymous URI as the value in
2811 the SOAP header element **/wsa:ReplyTo/wsa:Address**.

```
2812 <wsp:Policy wsu:Id="xs:ID" ?>
2813   <wsam:MCsupported/>
2814 </wsp:Policy> ?
```

2816 This policy assertion **MUST** only be used, if WS MakeConnection is supported by this endpoint (see
2817 chapter [6.2]). In this case, the value of **/wsa:ReplyTo/wsa:Address** **MUST** be the WS-MC
2818 anonymous URI template

²⁵ No further details defined here, it is left to implementations how to define appropriate count starting and reset points

2819 `http://docs.oasis-open.org/ws-rx/wsmc/200702/anonymous?id={unique-String}` .

2820 Following policy assertions MUST be bound to `wsdl11:ports` (WSDL 2.0: endpoints) or
 2821 `wsdl11:binding` policy subjects accepting messages for `MsgBox` access - these are the message
 2822 of type `MsgBoxFetchRequest`, `MsgBoxStatusListRequest`, `MsgBoxGetNextRequest` and
 2823 `MsgBoxCloseRequest`:

```
2824 <wsp:Policy wsu:Id="xs:ID" ?>
2825   <wsam:Addressing>
2826     <wsp:Policy>
2827       <wsam:AnonymousResponses/>
2828     <wsp:Policy>
2829   </wsam:Addressing>
2830 </wsp:Policy>
```

2831 This policy ascertains the use of WS-Addressing and that the endpoint requires request messages to
 2832 use response endpoint EPRs that carry an URI value of

2833 `"http://www.w3.org/2005/08/addressing/anonymous"`

2834 in the SOAP header element `/wsa:ReplyTo/wsa:Address`.

2835 10.4 WS Reliable Messaging Policy Assertions

2836 Support of WS Reliable Messaging is an optional feature of conformant implementations. If supported,
 2837 adequate policy assertions SHOULD be used to ascertain the details of reliable messaging exchange.
 2838 We refer to the specification WS Reliable Messaging Policy Assertions Version 1.1 [WSRMP] in this
 2839 point with no further profiling.

2840 10.5 MTOM Policy Assertion

2841 The SOAP Message Transmission Optimization Mechanism [MTOM] MUST be supported by
 2842 conformant implementations. The MTOM policy assertion MUST be attached to either a
 2843 `wsdl11:binding` or `wsdl11:port` endpoint policy subject. It is expressed as

```
2844 <wsp:Policy wsu:Id="xs:ID" ?>
2845   <wspmtom:OptimizedMimeSerialization/>
2846 </wsp:Policy ?>
```

2847 We refer to the specification draft [MTOMP].

2848 10.6 WS Security Profile and Policy Assertions

2849 10.6.1 Endpoint Policy Subject Assertions

2850 10.6.1.1 Symmetric Binding

2851 The symmetric binding assertion defines details of message protection by means of WS-Security
 2852 [WSS]. In both directions from the Initiator to the Recipient or his `MsgBox` instance and backwards the
 2853 same security tokens are used for transport level encryption and signature.

2854 According to [WSSP], this assertion SHOULD apply to the endpoint policy subject `wsdl11:binding`;
 2855 it MAY apply to operation policy subject `wsdl11:binding/wsdl11:operation`.

2856 Requirements outlined in this document for message security lead to following restrictions of overall
 2857 options defined by WS Security Policy (see [WSSP], chapter 7.4).

2858 As described in chapter [7.5, R0600], SAML Token issued by STS instances MUST be used, which
 2859 here leads to:

2860 **R1230** - This profiling restricts to the usage of `wssp:ProtectionToken`; distinct
 2861 `wssp:EncryptionToken` and `wssp:SignatureToken` MUST NOT be used.

2862 **10.6.1.2 Asymmetric Binding**

2863 For the asymmetric binding, public keys of X.509v3 certificates are used as security tokens. The
2864 support of this binding is OPTIONAL; it MUST be used in case of anonymous access is supported as
2865 described in chapter [6.2].

2866 According to [WSSP], this assertion SHOULD apply to the endpoint policy subject `wsd111:binding`;
2867 it MAY apply to operation policy subject `wsd111:binding/wsd111:operation`.

2868 Used certificates MUST have the according key usage set; R0610 and R0620 (see chapter [7.4])
2869 apply here and in addition:

2870 **R1240** - The node a message is targeted to MUST verify the validity of certificates used for
2871 encryption; in case a value other than valid at time of usage is stated, the message MUST
2872 be discarded and a fault MUST be generated.

2873 Fault 15: **EncryptionCertNotValid**

2874 [Code] Sender

2875 [Subcode] EncryptionCertNotValid

2876 [Reason] Encryption certificate not stated to be valid

2877 More information about the certificate validation results SHOULD be provided in the fault
2878 [Details] property in this case. It is strongly RECOMMENDED to log such faults to be able
2879 to detect possible security violation attacks.

2880 In the context with certificates used by a Recipient oder his MsgBox node as described in the chapter
2881 [10.2.1], the assertions `/wssp:RecipientEncryptionToken` and
2882 `/wssp:RecipientSignatureToken` SHOULD point to the according certificate entries in in the
2883 Recipients metadata file.

2884 **10.6.1.3 Transport Binding**

2885 The transport binding MAY be used in scenarios in which message protection and security correlation
2886 is provided by means other than WS-Security. We restrict to HTTPS here:

2887 **R1250** - HTTPS MUST be used, if message protection is provided by the underlying transport
2888 protocol.

2889 This assertion MUST apply to the endpoint policy subject `wsd111:binding`.

2890 **10.6.2 Message Policy Subject Assertions**

2891 [WSSP] offers policy statements for directions, which message parts must be present and which
2892 message parts have to be signed and encrypted. For the here presented profiling, assertions on the
2893 SOAP header and body block level are REQUIRED, assertions on element level according to [WSSP]
2894 MAY be used in addition.

2895 Following outlines only show the syntax of these assertions; following requirement applies:

2896 **R1260** - Concrete instances MUST enumerate the header and body blocks marked as mandatory
2897 for presence, to be signed and/or encrypted according to definitions made per message
2898 type in chapter [9, Constituents of OSCI Message Types].

2899 Required message parts policy assertion:

```
2900 <wsp:Policy>
2901   <wsp:ALL>
2902     <wssp:RequiredParts xmlns:wssp="..." ... >
2903       <wssp:Header Name="xs:NCName"? Namespace="xs:anyURI" ... /> +
2904     </wssp:RequiredParts>
2905   </wsp:ALL>
2906 </wsp:Policy>
```

2907 Signed message parts policy assertion:

```

2908 <wsp:Policy>
2909   <wsp:ALL>
2910     <wssp:SignedParts xmlns:wssp="..." ... >
2911       <wssp:Body />
2912       <wssp:Header Name="xs:NCName"? Namespace="xs:anyURI" ... /> +
2913     </wssp:SignedParts>
2914   </wsp:ALL>
2915 </wsp:Policy>

```

2916 **NOTE:** According to R1230, the SOAP body block always **MUST** be included in the transport
 2917 signature to ensure integrity of coherence with the message header block parts.

2918 Encrypted message parts policy assertion:

```

2919 <wsp:Policy>
2920   <wsp:ALL>
2921     <wssp:EncryptedParts xmlns:wssp="..." ... >
2922       <wssp:Body /> ?
2923       <wssp:Header Name="xs:NCName"? Namespace="xs:anyURI" ... /> *
2924     </wssp:EncryptedParts>
2925   </wsp:ALL>
2926 </wsp:Policy>

```

2927 If potentially unsecured network connections are used for message exchange, following requirement
 2928 applies:

2929 **R1270** - If the Content Data carried in the SOAP body is not encrypted end-to-end, the body block
 2930 **MUST** be transport encrypted.

2931 To include the required SOAP header blocks of the different OSCI message types, following
 2932 requirement applies:

2933 **R1280** - These policy assertions **MUST** be bound to the message policy subject:

2934 wsdl11:binding/wsdl11:operation/wsdl11:input

2935 respective

2936 wsdl11:binding/wsdl11:operation/wsdl11:output

2937 10.6.3 Algorithm Suite Assertions

2938 In the chapters [7.2.1 and 7.3.2], restrictions are defined to suitable cryptographic algorithms, which
 2939 leads to following restrictions²⁶:

2940 **R1290** - For the symmetric case, following restriction applies for the algorithm suite assertion:

```

2941 <wsp:Policy>
2942   <wssp:AlgorithmSuite>
2943     <wsp:Policy>
2944       ( <wssp:Basic256Sha256 ... /> |
2945         <wssp:Basic192Sha256 ... /> |
2946         <wssp:Basic128Sha256 ... /> |
2947         <wssp:TripleDesSha256 ... /> )
2948     </wsp:Policy>
2949   </wssp:AlgorithmSuite>
2950 </wsp:Policy>

```

2951 **R1300** - For the asymmetric case, following restriction applies for the algorithm suite assertion:

```

2952 <wsp:Policy>
2953   <wssp:AlgorithmSuite>

```

²⁶ As of today, there are not yet algorithm identifier assertions defined for SHA512 and RIPEMD160. As these are recommended algorithms, this will be aligned with the reusable OASIS TC and completed as soon as possible by corrigenda for this document.


```
2954 <wsp:Policy>
2955   ( <wssp:Basic256Sha256Rsa15 ... /> |
2956   <wssp:Basic192Sha256Rsa15 ... /> |
2957   <wssp:Basic128Sha256Rsa15 ... /> |
2958   <wssp:TripleDesSha256Rsa15 ... /> )
2959 </wsp:Policy>
2960 </wssp:AlgorithmSuite>
2961 </wsp:Policy>
```

2962 The scope of these assertions is defined by its containing assertion.

2963 **R1310** - Algorithm suite assertions **MUST** at least be included in assertions bound to the the
2964 endpoint policy subject **wsdl11:binding**. In addition, variations **MAY** be bound to
2965 subordinary policy subjects to express specific requirements.

2966 **11 Applying End-to-end Encryption and Digital Signatures** 2967 **on Content Data**

2968 Predominant for OSCI is exchange of data in an authentic, confidential manner with support for legal
2969 binding. Hence, functionalities are needed for Content Data end-to-end encryption and decryption,
2970 application of digital signatures to Content Data and signature validation.

2971 To ensure interoperability and conformance with the EC-Directive on Digital Signatures as well the
2972 German Signature Law and -Ordinance and underlying technical specifications, these optional
2973 functionalities – if provided – MUST be realized conformant to the "Common PKI Specifications for
2974 Interoperable Applications, Part 7: Signature API" [COMPKI]. This specification is a subset of the
2975 "eCard-API Framework" [eCardAPI], based on standards worked out by the OASIS Digital Signature
2976 Services Technical Committee [DSS].

2977 The Common PKI Signature API defines an XML interface for – among others – following functions:

- 2978 • SignRequest
- 2979 • VerifyRequest
- 2980 • EncryptRequest
- 2981 • DecryptRequest.

2982 API bindings are defined for C and Java; on base of the XML definitions the defined functions could
2983 also be realized as services provided by an OSCI Gateway implementation.

2984 To use the OSCI-feature of certificate validation on the message route, messages producing instances
2985 SHOULD supply certificates used for cryptographical operations on Content Data level in a structure
2986 described as "X.509-Token Container" in chapter [8.4.1]. This container must be carried in a message
2987 as custom SOAP header block.

2988 On the message consuming side, the resulting custom SOAP headers `/xkms:ValidateResponse`
2989 SHOULD be used to simplify signature verification, as the burden of connecting to CAs is delegated to
2990 specialized nodes on the message route. See chapter [8.4] for details.

2991 **12 Indices**

2992 **12.1 Tables**

2993	Table 1: Referenced Namespaces.....	9
2994	Table 2: Predefined business scenario types.....	14
2995	Table 3: Defined URIs for the WS Addressing Action element.....	17
2996	Table 4: Digest method: allowed algorithm identifiers.....	20
2997	Table 5: Signature method: allowed algorithm identifiers	20
2998	Table 6: Symmetric encryption algorithms	24
2999	Table 7: Security token types – support requirements.....	24
3000	Table 8: OSCI X.509-Token usages	81
3001	Table 9: SOAP/OSCI roles assigned to token usages.....	82
3002		

3003 **12.2 Pictures**

3004	Figure 1: Request Security Token Message	28
3005	Figure 2: Request Security Token, Body for Issue Request.....	29
3006	Figure 3: Request Security Token Response Message.....	31
3007	Figure 4: Request Security Token, Body for Issue Response	32
3008	Figure 5: SAML 2.0 Assertion constituents	33
3009	Figure 6: RST for OneTimeToken	37
3010	Figure 7: RSTR for OneTimeToken	38
3011	Figure 8: osci:Request header and body block assembly.....	68
3012	Figure 9: osci:Response header and body block assembly.....	70
3013	Figure 10: MsgBoxFetchRequest header and body block assembly.....	72
3014	Figure 11: MsgBoxStatusListRequest header and body block assembly	73
3015	Figure 12: MsgBoxResponse header and body block assembly	74
3016	Figure 13: MsgBoxGetNextRequest header and body block assembly.....	75
3017	Figure 14: MsgBoxClose header and body block assembly	76

3018 **12.3 OSCI specific faults**

3019	Fault 1: AddrWrongTypeOfBusinessScenario.....	15
3020	Fault 2: AddrWrongActionURI	17
3021	Fault 3: AuthnCertNotValid.....	25
3022	Fault 4: AuthnCertInvalidKeyUsage	25
3023	Fault 5: AuthnSecurityLevelInsufficient	27

3024	Fault 6: AuthnTokenFormalMismatch	35
3025	Fault 7: MsgBoxRequestWrongReference.....	51
3026	Fault 8: QualTSPServiceNotAvailable.....	57
3027	Fault 9: MsgBodyDecryptionError	58
3028	Fault 10: SignatureOfReceiptInvalid.....	60
3029	Fault 11: SignatureOfValidateResultInvalid.....	66
3030	Fault 12: MsgHeaderStructureSchemaViolation	66
3031	Fault 13: MsgSizeLimitExceeded	85
3032	Fault 14: MsgFrequencyLimitExceeded	85
3033	Fault 15: EncryptionCertNotValid	87
3034		

3035 **12.4 Listings**

3036	Listing 1: ExampleEndpointOSCIPolicy.xml.....	104
3037	Listing 2: Example XML Signature	106
3038		

3039 13 References

3040 13.1 Normative

- 3041 [AlgCat] Bekanntmachung zur elektronischen Signatur nach dem Signaturgesetz und der
3042 Signaturverordnung (Übersicht über geeignete Algorithmen), Bundesnetzagentur für
3043 Elektrizität, Gas, Telekommunikation, Post und Eisenbahnen, 17. November 2008,
3044 <http://www.bundesnetzagentur.de/media/archive/14953.pdf>
- 3045 [COMPKI] Common PKI Specifications for interoperable Applications, Version 2.0, 20 January
3046 2009; [http://www.common-pki.org/uploads/media/Common-](http://www.common-pki.org/uploads/media/Common-PKI_v2.0.pdf)
3047 [PKI_v2.0.pdf](http://www.common-pki.org/uploads/media/Common-PKI_v2.0.pdf)
- 3048 [eCardAPI] Das eCard-API-Framework (BSI TR-03112). Version 1.0, Federal Office for
3049 Information Security (Bundesamt für Sicherheit in der Informationstechnik), March
3050 2008, <http://www.bsi.de/literat/tr/tr03112/index.htm>
- 3051 [DSS] Digital Signature Service Core - Protocols, Elements, and Bindings Version 1.0,
3052 OASIS Standard, 11 April 2007; [http://www.oasis-](http://www.oasis-open.org/specs/index.php#dssv1.0)
3053 [open.org/specs/index.php#dssv1.0](http://www.oasis-open.org/specs/index.php#dssv1.0)
- 3054 [MTOM] SOAP Message Transmission Optimization Mechanism, W3C Recommendation 25
3055 January 2005, <http://www.w3.org/TR/soap12-mtom/>
- 3056 [MTOMP] MTOM Policy Assertion 1.1, W3C Working Draft 18 September 2007,
3057 <http://www.w3.org/TR/soap12-mtom-policy/>
- 3058 [PKCS#1] B. Kaliski, J. Staddon: PKCS #1: RSA Cryptography Specifications – Version 2.0,
3059 IETF RFC 2437, The Internet Society October 1998,
3060 <http://www.ietf.org/rfc/rfc2437.txt>
- 3061 [RFC2119] S. Bradner, **Key words for use in RFCs to Indicate Requirement**
3062 **Levels**, RFC 2119, Harvard University, March 1997,
3063 <http://www.ietf.org/rfc/rfc2119.txt>
- 3064 [RFC2396] T. Berners-Lee, R. Fielding, U.C. Irvine, L. Masiner, Uniform Resource Identifiers
3065 (URI): Generic Syntax, RFC 2396, The Internet Society 1998;
3066 <http://www.ietf.org/rfc/rfc2396.txt>
- 3067 [RFC3161] D. Pinkas, R. Zuccherato, Time-Stamp Protocol (TSP), IETF RFC 1661,
3068 <http://www.ietf.org/rfc/rfc3161.txt>
- 3069 [RFC4122] A Universally Unique Identifier (UUID) URN Namespace, The Internet Engineering
3070 Task Force July 2005, <http://www.ietf.org/rfc/rfc4122.txt>
- 3071 [SAFE] S.A.F.E. (Secure Access to Federated e-Justice/e-Government) / D.I.M. (Distributed
3072 Identity Management), Unterarbeitsgruppe SAFE der BLK Arbeitsgruppe ITStandards
3073 in der Justiz, April 2008, [http://www.justiz.de/ERV/Grob-](http://www.justiz.de/ERV/Grob-_und_Feinkonzept/index.php)
3074 [_und_Feinkonzept/index.php](http://www.justiz.de/ERV/Grob-_und_Feinkonzept/index.php)
- 3075 [SAMLAC] Authentication Context for the OASIS Security Assertion Markup Language (SAML)
3076 V2.0, OASIS Standard, 15 March 2005, [http://docs.oasis-](http://docs.oasis-open.org/security/saml/v2.0/saml-authn-context-2.0-os.pdf)
3077 [open.org/security/saml/v2.0/saml-authn-context-2.0-os.pdf](http://docs.oasis-open.org/security/saml/v2.0/saml-authn-context-2.0-os.pdf)
- 3078 [SAML2] Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML)
3079 V2.0, OASIS Standard, 15 March 2005; [http://docs.oasis-](http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf)
3080 [open.org/security/saml/v2.0/saml-core-2.0-os.pdf](http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf)
- 3081 [SOAP12] SOAP Version 1.2 Part 1: Messaging Framework (Second Edition), W3C
3082 Recommendation 27 April 2007, <http://www.w3.org/TR/soap12-part1/>

3083	[WSA]	Web Services Addressing 1.0 - Core, W3C Recommendation 9 May 2006,
3084		http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/
3085	[WSAM]	Web Services Addressing 1.0 - Metadata, W3C Proposed Recommendation 31 July
3086		2007, http://www.w3.org/TR/ws-addr-metadata/
3087	[WSASOAP]	Web Services Addressing 1.0 – SOAP Binding, W3C Recommendation 9 May 2006,
3088		http://www.w3.org/TR/2006/REC-ws-addr-soap-20060509/
3089	[WSAW]	Web Services Addressing 1.0 – WSDL Binding, W3C Candidate Recommendation 29
3090		May 2006, http://www.w3.org/TR/ws-addr-wsdl/
3091	[WSDL20]	Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language,
3092		W3C Recommendation 26 June 2007, http://www.w3.org/TR/wsdl20/
3093	[WSDL11]	Web Services Description Language (WSDL) 1.1: W3C Note 15 March 2001,
3094		http://www.w3.org/TR/2001/NOTE-wsdl-20010315
3095	[WSDLA]	Web Services Description Language (WSDL) Version 2.0 Part 2: Adjuncts, W3C
3096		Recommendation 26 June 2007, http://www.w3.org/TR/2007/REC-wsdl20-adjuncts-20070626/
3097		
3098	[WSF]	Web Services Federation Language (WS-Federation), Version 1.1, December 2006,
3099		http://specs.xmlsoap.org/ws/2006/12/federation/ws-
3100		federation.pdf
3101	[WSI-Basic]	WS-I Basic Profile 2.0, Working Group Draft, 2007-10-25, WEB SERVICES
3102		INTEROPERABILITY ORGANIZATION, http://www.ws-
3103		i.org/Profiles/BasicProfile-2_0(WGD).html
3104	[WSI-BP11]	WS-I Basic Profile 1.1, Final Material, 2006-04-10, WEB SERVICES
3105		INTEROPERABILITY ORGANIZATION, http://www.ws-
3106		i.org/Profiles/BasicProfile-1.1.html
3107	[WSI-BSP11]	WS-I Basic Security Profile Version 1.1, Working Group Approval Draft, 2007-02-20,
3108		WEB SERVICES INTEROPERABILITY ORGANIZATION, http://www.ws-
3109		i.org/Profiles/BasicSecurityProfile-1.1.html
3110	[WSMC]	Web Services Make Connection (WS MakeConnection), Version 1.0, OASIS
3111		Standard, 14 June 2007, http://docs.oasis-open.org/ws-
3112		rx/wsmc/200702/wsmc-1.0-spec-os-01.pdf
3113	[WSPA]	Web Services Policy 1.5 - Attachment, W3C Recommendation, 4 September 2007;
3114		http://www.w3.org/TR/2007/REC-ws-policy-attach-20070904/
3115	[WSPF]	Web Services Policy 1.5 - Framework, W3C Recommendation, 4 September 2007;
3116		http://www.w3.org/TR/2007/REC-ws-policy-20070904/
3117	[WSRM]	Web Services Reliable Messaging (WS-ReliableMessaging) Version 1.1, OASIS
3118		Standard Specification incorporating approved Errata, 07 January 2008,
3119		http://docs.oasis-open.org/ws-rx/wsrmp/200702/wsrmp-1.1-spec-os-
3120		01-e1.pdf
3121	[WSRMP]	Web Services Reliable Messaging Policy Assertion (WS-RM Policy) Version 1.1
3122		OASIS Standard Specification incorporating approved Errata, 07 January 2008,
3123		http://docs.oasis-open.org/ws-rx/wsrmp/200702/wsrmp-1.1-spec-
3124		os-01-e1.pdf
3125	[WSS]	Web Services Security: SOAP Message Security 1.1, OASIS Standard Specification,
3126		1 February 2006, http://www.oasis-
3127		open.org/committees/download.php/16790/wss-v1.1-spec-os-
3128		SOAPMessageSecurity.pdf

- 3129 [WSS10] Web Services Security: SOAP Message Security 1.0 (WS-Security 200), OASIS
3130 Standard 200401, March 2004, [http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-soap-message-security-
1.0.pdf](http://docs.oasis-
3131 open.org/wss/2004/01/oasis-200401-wss-soap-message-security-
3132 1.0.pdf)
- 3133 [WSSC] Web Services Secure Conversation 1.3, OASIS Standard, 1 March 2007,
3134 [http://docs.oasis-open.org/ws-sx/ws-
secureconversation/200512/ws-secureconversation-1.3-os.pdf](http://docs.oasis-open.org/ws-sx/ws-
3135 secureconversation/200512/ws-secureconversation-1.3-os.pdf)
- 3136 [WSSP] WS-SecurityPolicy 1.2, OASIS Standard 1 July 2007, [http://docs.oasis-
open.org/ws-sx/ws-securitypolicy/200702/ws-securitypolicy-1.2-
spec-os.pdf](http://docs.oasis-
3137 open.org/ws-sx/ws-securitypolicy/200702/ws-securitypolicy-1.2-
3138 spec-os.pdf)
- 3139 [WSSKERB] Web Services Security Kerberos Token Profile 1.1, OASIS Standard Specification,
3140 incorporating Approved Errata, 1 November 2006, [http://docs.oasis-
open.org/wss/v1.1/wss-v1.1-spec-errata-os-
KerberosTokenProfile.pdf](http://docs.oasis-
3141 open.org/wss/v1.1/wss-v1.1-spec-errata-os-
3142 KerberosTokenProfile.pdf)
- 3143 [WSSSAML] Web Services Security SAML Token Profile 1.1, OASIS Standard Specification
3144 incorporating Approved Errata, 1 November 2006, [http://docs.oasis-
open.org/wss/v1.1/wss-v1.1-spec-errata-os-SAMLTOKENPROFILE.pdf](http://docs.oasis-
3145 open.org/wss/v1.1/wss-v1.1-spec-errata-os-SAMLTOKENPROFILE.pdf)
- 3146 [WSSUSER] Web Services Security Username Token Profile 1.1, OASIS Standard Specification, 1
3147 February 2006, [http://www.oasis-
open.org/committees/download.php/16782/wss-v1.1-spec-os-
UsernameTokenProfile.pdf](http://www.oasis-
3148 open.org/committees/download.php/16782/wss-v1.1-spec-os-
3149 UsernameTokenProfile.pdf)
- 3150 [WSSX509] Web Services Security X.509 Certificate Token Profile 1.1, OASIS Standard
3151 Specification, incorporating Approved Errata, 1 November 2006,
3152 [http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-errata-os-
x509TokenProfile.pdf](http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-errata-os-
3153 x509TokenProfile.pdf)
- 3154 [WST] WS-Trust 1.3, OASIS Standard, 19 March 2007, [http://docs.oasis-
open.org/ws-sx/ws-trust/200512/ws-trust-1.3-os.pdf](http://docs.oasis-
3155 open.org/ws-sx/ws-trust/200512/ws-trust-1.3-os.pdf)
- 3156 [XAdES] European Telecommunications Standards Institute. ETSI TS 101 903: XML Advanced
3157 Electronic Signatures, V1.3.2 2006-03;
3158 http://webapp.etsi.org/action/PU/20060307/ts_101903v010302p.pdf
3159 .
- 3160 [XENC] World Wide Web Consortium. XML Encryption Syntax and Processing, W3C
3161 Recommendation, 10.12.2002;
3162 <http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/>
- 3163 [XKMS] XML Key Management Specification (XKMS 2.0) v2, W3C Recommendation, 28 June
3164 2005, <http://www.w3.org/TR/2005/REC-xkms2-20050628/>
- 3165 [XKMSEU] PEPPOL XKMS v2 Interface Specification, Version 1.2, PEPPOL WP1 2009-04-30,
3166 copy at [http://www.osci.eu/transport/osci2/specification/PEPPOL_D1.1_v1.2_XKMS
InterfaceSpecification](http://www.osci.eu/transport/osci2/specification/PEPPOL_D1.1_v1.2_XKMS
3167 InterfaceSpecification)
- 3168 [XMLDSIG] World Wide Web Consortium. XML-Signature Syntax and Processing (Second
3169 Edition), W3C Recommendation, 10 June 2008;
3170 <http://www.w3.org/TR/xmlldsig-core/>
- 3171 [XMLSchema] World Wide Web Consortium. XML Schema, Parts 0, 1, and 2 (Second Edition). W3C
3172 Recommendation, 28 October 2004; <http://www.w3.org/TR/xmlschema-0/>,
3173 <http://www.w3.org/TR/xmlschema-1/>, and
3174 <http://www.w3.org/TR/xmlschema-2/>
- 3175 [XML 1.0] World Wide Web Consortium. Extensible Markup Language (XML) 1.0 (Fourth
3176 Edition), T. Bray, J. Paoli, C. M. Sperberg-McQueen, and E. Maler, Editors. 10

- 3177 February 1998, revised 16 August 2006; <http://www.w3.org/TR/2006/REC-xml-20060816/>
3178
- 3179 [XPATH 1.0] W3C Recommendation, "XML Path Language (XPath) Version 1.0," 16
3180 November 1999; <http://www.w3.org/TR/xpath>

3181 13.2 Informative

- 3182 [WSFED] Web Services Federation Language (WS-Federation), Version 1.2, latest TC/Editor
3183 Draft see: [http://www.oasis-](http://www.oasis-open.org/committees/documents.php?wg_abbrev=wsfed)
3184 [open.org/committees/documents.php?wg_abbrev=wsfed](http://www.oasis-open.org/committees/documents.php?wg_abbrev=wsfed)
- 3185 [WSMEX] Web Services Metadata Exchange, Version 1.1, W3C Member Submission 13 August
3186 2008, [http://www.w3.org/Submission/2008/SUBM-WS-](http://www.w3.org/Submission/2008/SUBM-WS-MetadataExchange-20080813/)
3187 [MetadataExchange-20080813/](http://www.w3.org/Submission/2008/SUBM-WS-MetadataExchange-20080813/)

3188 Appendix A. Schema

3189 OSCI Transport 2.0 Schema

```

3190 <?xml version="1.0" encoding="UTF-8"?>
3191 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
3192 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
3193 xmlns:wsa="http://www.w3.org/2005/08/addressing"
3194 xmlns:osci="http://www.osci.eu/ws/2008/05/transport" xmlns:wsu="http://docs.oasis-
3195 open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
3196 xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
3197 xmlns:wsp="http://www.w3.org/ns/ws-policy" xmlns:wsse="http://docs.oasis-
3198 open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
3199 targetNamespace="http://www.osci.eu/ws/2008/05/transport"
3200 elementFormDefault="qualified" attributeFormDefault="unqualified">
3201   <!--OSCI Transport Version 2.0 schema - last edited by Joerg Apitzsch/bos as of
3202   2009-06-04-->
3203   <!--xs:import namespace="http://www.w3.org/2005/08/addressing" schemaLocation="ws-
3204   addr.xsd"/-->
3205   <xs:import namespace="http://www.w3.org/2005/08/addressing"
3206   schemaLocation="http://www.w3.org/2006/03/addressing/ws-addr.xsd"/>
3207   <!--xs:import namespace="http://www.w3.org/2000/09/xmldsig#"
3208   schemaLocation="xmldsig-core-schema.xsd"/-->
3209   <xs:import namespace="http://www.w3.org/2000/09/xmldsig#"
3210   schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-schema.xsd"/>
3211   <!--xs:import namespace="http://www.w3.org/2003/05/soap-envelope"
3212   schemaLocation="soap-envelope.xsd"/-->
3213   <xs:import namespace="http://www.w3.org/2003/05/soap-envelope"
3214   schemaLocation="http://www.w3.org/2003/05/soap-envelope"/>
3215   <!--xs:import namespace="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
3216   wssecurity-utility-1.0.xsd" schemaLocation="oasis-200401-wss-wssecurity-utility-
3217   1.0.xsd"/-->
3218   <xs:import namespace="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
3219   wssecurity-utility-1.0.xsd" schemaLocation="http://docs.oasis-
3220   open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"/>
3221   <!--xs:import namespace="http://www.w3.org/ns/ws-policy" schemaLocation="ws-
3222   policy.xsd"/-->
3223   <xs:import namespace="http://www.w3.org/ns/ws-policy"
3224   schemaLocation="http://www.w3.org/2007/02/ws-policy.xsd"/>
3225   <!--xs:import namespace="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
3226   wssecurity-secext-1.0.xsd" schemaLocation="oasis-200401-wss-wssecurity-secext-
3227   1.0.xsd"/-->
3228   <xs:import namespace="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
3229   wssecurity-secext-1.0.xsd" schemaLocation="http://docs.oasis-
3230   open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"/>
3231   <!--Roles of OSCI (SOAP) nodes (informational, not referenced by other
3232   definitions) -->
3233   <xs:simpleType name="Role">
3234     <xs:list itemType="xs:anyURI"/>
3235   </xs:simpleType>
3236   <!--WSA-Extension: BusinessScenarioType-->
3237   <xs:element name="TypeOfBusinessScenario" type="xs:anyURI"/>
3238   <!--General header-part of OSCI messages: timestamps-->
3239   <xs:complexType name="MsgTimeStampsType">
3240     <xs:sequence>
3241       <xs:element name="ObsoleteAfter" type="xs:date" minOccurs="0">
3242         <xs:annotation>
3243           <xs:documentation>Date, when this message is obsolete; may be set by
3244   Initiator</xs:documentation>
3245         </xs:annotation>
3246       </xs:element>
3247       <xs:element name="Delivery" type="xs:dateTime" minOccurs="0">
3248         <xs:annotation>
3249           <xs:documentation>Time of entry in a Recipient MsgBox</xs:documentation>
3250         </xs:annotation>
3251       </xs:element>
3252       <xs:element name="InitialFetch" type="xs:dateTime" minOccurs="0">

```

```

3253         <xs:annotation>
3254         <xs:documentation>Time of first comitted fetch from MsgBox by the
3255 Recipient</xs:documentation>
3256         </xs:annotation>
3257     </xs:element>
3258     <xs:element name="Reception" type="xs:dateTime" minOccurs="0">
3259         <xs:annotation>
3260         <xs:documentation>Reception Time set by the Recipient</xs:documentation>
3261         </xs:annotation>
3262     </xs:element>
3263 </xs:sequence>
3264 <xs:attribute ref="wsu:Id" use="required"/>
3265 </xs:complexType>
3266 <xs:element name="MsgTimeStamps" type="osci:MsgTimeStampsType"/>
3267 <!--Types and Elements for MsgBox request/responses-->
3268 <xs:annotation>
3269 <xs:documentation>Template for MsgBox-Requests</xs:documentation>
3270 </xs:annotation>
3271 <xs:complexType name="MsgBoxRequestType">
3272     <xs:sequence>
3273         <xs:element ref="wsa:EndpointReference"/>
3274         <xs:element ref="osci:MsgSelector" minOccurs="0"/>
3275     </xs:sequence>
3276 </xs:complexType>
3277 <xs:complexType name="MsgBoxResponseType">
3278     <xs:choice>
3279         <xs:element name="NoMessageAvailable">
3280             <xs:complexType>
3281                 <xs:attribute name="reason" type="xs:QName"/>
3282             </xs:complexType>
3283         </xs:element>
3284         <xs:element name="ItemsPending" type="xs:nonNegativeInteger"/>
3285     </xs:choice>
3286     <xs:attribute ref="osci:MsgBoxRequestID" use="required"/>
3287     <xs:attribute ref="wsu:Id"/>
3288 </xs:complexType>
3289 <xs:complexType name="MsgAttributeListType">
3290     <xs:sequence>
3291         <xs:element ref="wsa:MessageID"/>
3292         <xs:element ref="wsa:RelatesTo" minOccurs="0" maxOccurs="unbounded"/>
3293         <xs:element ref="wsa:From"/>
3294         <xs:element name="BusinessScenarioType" type="xs:QName"/>
3295         <xs:element ref="wsa:Action"/>
3296         <xs:element name="MsgSize" type="xs:int"/>
3297         <!--xs:element ref="osci:MsgTimeStamps"/-->
3298         <xs:element name="ObsoleteAfterDate" type="xs:date"/>
3299         <xs:element name="DeliveryTime" type="xs:dateTime"/>
3300         <xs:element name="InitialFetchedTime" type="xs:dateTime"/>
3301     </xs:sequence>
3302 </xs:complexType>
3303 <xs:attribute name="MsgBoxRequestID" type="xs:anyURI"/>
3304 <xs:element name="MsgSelector">
3305     <xs:complexType>
3306         <xs:sequence minOccurs="0">
3307             <xs:element ref="wsa:MessageID" minOccurs="0" maxOccurs="unbounded"/>
3308             <xs:element ref="wsa:RelatesTo" minOccurs="0" maxOccurs="unbounded"/>
3309             <xs:element name="MsgBoxEntryFrom" type="xs:dateTime" minOccurs="0"/>
3310             <xs:element name="MsgBoxEntryTo" type="xs:dateTime" minOccurs="0"/>
3311             <xs:element name="Extension" type="xs:anyType" minOccurs="0"/>
3312         </xs:sequence>
3313         <xs:attribute name="newEntry" type="xs:boolean"/>
3314     </xs:complexType>
3315 </xs:element>
3316 <xs:element name="MsgStatusList" type="osci:MsgStatusListType"/>
3317 <xs:complexType name="MsgStatusListType">
3318     <xs:sequence>
3319         <xs:element name="MsgAttributes" type="osci:MsgAttributeListType"
3320 maxOccurs="unbounded"/>
3321     </xs:sequence>
3322 </xs:complexType>

```

```

3323 <xs:element name="MsgBoxFetchRequest" type="osci:MsgBoxRequestType"/>
3324 <xs:element name="MsgBoxStatusListRequest"
3325 type="osci:MsgBoxStatusListRequestType"/>
3326 <xs:complexType name="MsgBoxStatusListRequestType">
3327 <xs:complexContent>
3328 <xs:extension base="osci:MsgBoxRequestType">
3329 <xs:attribute name="maxListItems" type="xs:positiveInteger"/>
3330 </xs:extension>
3331 </xs:complexContent>
3332 </xs:complexType>
3333 <xs:element name="MsgBoxResponse" type="osci:MsgBoxResponseType"/>
3334 <xs:element name="MsgBoxGetNextRequest" type="osci:MsgBoxGetNextRequestType"/>
3335 <xs:complexType name="MsgBoxGetNextRequestType">
3336 <xs:sequence>
3337 <xs:element ref="wsa:EndpointReference"/>
3338 <!--
3339 <xs:element name="LastMsgReceived" type="wsa:AttributedURIType" minOccurs="0"
3340 maxOccurs="unbounded"/>
3341 -->
3342 </xs:sequence>
3343 <xs:attribute ref="osci:MsgBoxRequestID" use="required"/>
3344 </xs:complexType>
3345 <xs:element name="MsgBoxClose" type="osci:MsgBoxCloseType"/>
3346 <xs:complexType name="MsgBoxCloseType">
3347 <xs:attribute ref="osci:MsgBoxRequestID" use="required"/>
3348 <!--
3349 <xs:sequence minOccurs="0">
3350 <xs:element name="LastMsgReceived" type="wsa:AttributedURIType" minOccurs="0"
3351 maxOccurs="unbounded"/>
3352 </xs:sequence>
3353 -->
3354 </xs:complexType>
3355 <!--Types and Elements for Receipt- and Notification Handling-->
3356 <xs:attribute name="qualTSPForReceipt" type="xs:boolean" default="false"/>
3357 <xs:attribute name="echoRequest" type="xs:boolean" default="false"/>
3358 <xs:complexType name="ReceiptDemandType">
3359 <xs:sequence>
3360 <xs:element ref="wsa:ReplyTo"/>
3361 </xs:sequence>
3362 <xs:attribute ref="wsu:Id" use="required"/>
3363 <xs:attribute ref="s12:role"/>
3364 <xs:attribute ref="osci:qualTSPForReceipt"/>
3365 <xs:attribute ref="osci:echoRequest"/>
3366 </xs:complexType>
3367 <xs:element name="DeliveryReceiptDemand" type="osci:DeliveryReceiptDemandType"/>
3368 <xs:element name="ReceptionReceiptDemand" type="osci:ReceptionReceiptDemandType"/>
3369 <xs:complexType name="ReceiptInfoType">
3370 <xs:sequence>
3371 <xs:element ref="wsa:MessageID"/>
3372 <xs:element ref="osci:MsgTimeStamps"/>
3373 <xs:element ref="wsa:RelatesTo" minOccurs="0"/>
3374 <xs:element name="To" type="wsa:EndpointReferenceType"/>
3375 <xs:element ref="wsa:From" minOccurs="0"/>
3376 <xs:element ref="wsa:ReplyTo"/>
3377 <xs:element name="RequestEcho" type="xs:base64Binary" minOccurs="0"/>
3378 </xs:sequence>
3379 <xs:attribute ref="wsu:Id" use="required"/>
3380 </xs:complexType>
3381 <xs:complexType name="DeliveryReceiptDemandType">
3382 <xs:complexContent>
3383 <xs:restriction base="osci:ReceiptDemandType">
3384 <xs:sequence>
3385 <xs:element ref="wsa:ReplyTo"/>
3386 </xs:sequence>
3387 <xs:attribute ref="s12:role" fixed="http://www.w3.org/2003/05/soap-
3388 envelope/role/next"/>
3389 </xs:restriction>
3390 </xs:complexContent>
3391 </xs:complexType>
3392 <xs:complexType name="ReceptionReceiptDemandType">

```

```

3393     <xs:complexContent>
3394       <xs:restriction base="osci:ReceiptDemandType">
3395         <xs:sequence>
3396           <xs:element ref="wsa:ReplyTo"/>
3397         </xs:sequence>
3398         <xs:attribute ref="s12:role" fixed="http://www.w3.org/2003/05/soap-
3399 envelope/role/ultimateReceiver"/>
3400       </xs:restriction>
3401     </xs:complexContent>
3402 </xs:complexType>
3403 <xs:complexType name="DeliveryReceiptType">
3404   <xs:sequence>
3405     <xs:element name="ReceiptInfo">
3406       <xs:complexType>
3407         <xs:complexContent>
3408           <xs:extension base="osci:ReceiptInfoType">
3409             <xs:attribute name="ReceiptIssuerRole" use="required">
3410               <xs:simpleType>
3411                 <xs:restriction base="xs:anyURI">
3412                   <xs:enumeration
3413 value="http://www.osci.eu/2008/transport/roles/MsgBox"/>
3414                   <xs:enumeration
3415 value="http://www.osci.eu/2008/transport/roles/Recipient"/>
3416                 </xs:restriction>
3417               </xs:simpleType>
3418             </xs:attribute>
3419           </xs:extension>
3420         </xs:complexContent>
3421       </xs:complexType>
3422     </xs:element>
3423     <xs:element ref="ds:Signature"/>
3424   </xs:sequence>
3425   <xs:attribute ref="wsu:Id" use="required"/>
3426 </xs:complexType>
3427 <xs:element name="DeliveryReceipt" type="osci:DeliveryReceiptType"/>
3428 <xs:complexType name="ReceptionReceiptType">
3429   <xs:sequence>
3430     <xs:element name="ReceiptInfo" type="osci:ReceiptInfoType"/>
3431     <xs:element ref="ds:Signature"/>
3432   </xs:sequence>
3433   <xs:attribute ref="wsu:Id"/>
3434 </xs:complexType>
3435 <xs:element name="ReceptionReceipt" type="osci:ReceptionReceiptType"/>
3436 <xs:complexType name="FetchedNotificationDemandType">
3437   <xs:sequence>
3438     <xs:element ref="wsa:ReplyTo"/>
3439   </xs:sequence>
3440   <xs:attribute ref="s12:role"
3441 default="http://www.osci.eu/2008/transport/roles/MsgBox"/>
3442   <xs:attribute ref="wsu:Id"/>
3443 </xs:complexType>
3444 <xs:element name="FetchedNotificationDemand"
3445 type="osci:FetchedNotificationDemandType"/>
3446 <xs:complexType name="FetchedNotificationType">
3447   <xs:sequence>
3448     <xs:element name="FetchedTime" type="xs:dateTime"/>
3449     <xs:element ref="wsa:MessageID"/>
3450     <xs:element ref="wsa:To"/>
3451     <xs:element ref="wsa:From"/>
3452   </xs:sequence>
3453 </xs:complexType>
3454 <xs:element name="FetchedNotification" type="osci:FetchedNotificationType"/>
3455 <!--Extensions for Key usage context-->
3456 <xs:complexType name="X509TokenContainerType">
3457   <xs:sequence maxOccurs="unbounded">
3458     <xs:element ref="osci:X509TokenInfo"/>
3459   </xs:sequence>
3460   <xs:attribute name="validateCompleted" type="xs:boolean" default="false"/>
3461   <xs:attribute ref="wsu:Id"/>
3462   <xs:attribute ref="s12:role"/>

```

```

3463 </xs:complexType>
3464 <xs:element name="X509TokenContainer" type="osci:X509TokenContainerType"/>
3465 <xs:element name="X509TokenInfo">
3466   <xs:complexType>
3467     <xs:sequence>
3468       <xs:element ref="ds:X509Data"/>
3469       <xs:element name="TokenApplication" maxOccurs="unbounded">
3470         <xs:complexType>
3471           <xs:sequence>
3472             <xs:element name="TimeInstant" type="xs:dateTime"/>
3473             <xs:element name="MsgItemRef" type="xs:IDREF" minOccurs="0"/>
3474           </xs:sequence>
3475           <xs:attribute name="validateResultRef" type="xs:IDREF"/>
3476           <xs:attribute name="ocspNoCache" type="xs:boolean"/>
3477         </xs:complexType>
3478       </xs:element>
3479     </xs:sequence>
3480     <xs:attribute name="validated" type="xs:boolean" default="false"/>
3481     <xs:attribute name="Id" type="xs:ID" use="required"/>
3482     <!-- RFC 3280 for KeyUsage with Extentions Attribute Certificate and usage
3483 for Authentication -->
3484   </xs:complexType>
3485   <!--OSCI Policy Asserstions-->
3486   <!--Policy qualified Timestamp Servcie available-->
3487 </xs:element>
3488 <!--Poliy Assertion carrying Endpoints X509Certificates-->
3489 <xs:element name="X509CertificateAssertion">
3490   <xs:complexType>
3491     <xs:sequence>
3492       <xs:element ref="wsp:All"/>
3493     </xs:sequence>
3494   </xs:complexType>
3495 </xs:element>
3496 <!--Policy, when qualified TSP service can be requested form this node-->
3497 <xs:element name="QualTspAssertion">
3498   <xs:complexType>
3499     <xs:attribute name="PolicyRef" type="xs:anyURI"/>
3500   </xs:complexType>
3501 </xs:element>
3502 <!--Policy if and how MsgTimeStamps:OsoleteAfter is handled-->
3503 <xs:element name="ObsoleteAfterAssertion">
3504   <xs:complexType>
3505     <xs:sequence>
3506       <xs:element name="MsgRetainDays" type="xs:positiveInteger"/>
3507       <xs:element name="WarningBeforeMsgObsolete" type="xs:positiveInteger"
3508 minOccurs="0"/>
3509     </xs:sequence>
3510     <xs:attribute name="PolicyRef" type="xs:anyURI"/>
3511   </xs:complexType>
3512 </xs:element>
3513 <!--Poliy for MakeConnection: Response Retention Days-->
3514 <xs:element name="MsgRetainDays" type="xs:positiveInteger"/>
3515 <!--Enumeration for possible X509 Token Usages-->
3516 <xs:attribute name="TokenUsage">
3517   <xs:simpleType>
3518     <xs:restriction base="xs:anyURI">
3519       <xs:enumeration
3520 value="http://www.osci.eu/common/names/TokenUsage/e2eContentEncryption"/>
3521       <xs:enumeration
3522 value="http://www.osci.eu/common/names/TokenUsage/TransportEncryption"/>
3523       <xs:enumeration
3524 value="http://www.osci.eu/common/names/TokenUsage/ReceiptSigning"/>
3525       <xs:enumeration
3526 value="http://www.osci.eu/common/names/TokenUsage/TSPSigning"/>
3527     </xs:restriction>
3528   </xs:simpleType>
3529 </xs:attribute>
3530 <!--Opaque Body Type - not used-->
3531 <!--Policy maximum accepted Message size and Frequency per hour-->
3532 <xs:element name="AcceptedMsgLimits">

```

```
3533     <xs:complexType>
3534       <xs:sequence>
3535         <xs:element name="MaxSize" type="xs:positiveInteger"/>
3536         <xs:element name="MaxPerHour" type="xs:positiveInteger"/>
3537       </xs:sequence>
3538     </xs:complexType>
3539 </xs:element>
3540 <xs:complexType name="MessageBody">
3541   <xs:sequence>
3542     <xs:any namespace="##any" minOccurs="0" maxOccurs="unbounded"/>
3543   </xs:sequence>
3544 </xs:complexType>
3545 </xs:schema>
```

3546 Appendix B. Example: OSCI Endpoint Metadata 3547 Instance

3548 This example is a policy instance of the schema explained in chapter [10.2]. The encryption and
3549 signature certificates exposed in this policy are addressed by URI references; according to [WSS]
3550 they could also be embedded in this policy file itself in base64Binary format.

3551 This endpoint exposes different encryption and signature certificates for the MsgBox and Recipient /
3552 UltimateRecipient instances. The [ObsoleteAfter] property is handled by this endpoint, while a service
3553 for qualified timestamps is not offered.

3554 For readability of policies used in the OSCI Transport context, it is strongly RECOMMENDED
3555 generally to use the `wsu:id` attribute values highlighted **bold** in this example, as these policy parts
3556 and certificates are referenced by other policies or service instances like a STS (i.e., the latter needs
3557 access to the endpoint encryption certificate for appropriate SAML token encryption).
3558

```

3559 <?xml version="1.0" encoding="UTF-8"?>
3560 <!-- OSCI specific endpoint metadata / policies -->
3561 <wsp:Policy Name="ExampleEndpointOSCIPolicy"
3562 xsi:schemaLocation="http://schemas.xmlsoap.org/ws/2004/09/policy
3563 http://schemas.xmlsoap.org/ws/2004/09/policy/ws-policy.xsd"
3564 xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
3565 utility-1.0.xsd" xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
3566 xmlns:wspmtom="http://schemas.xmlsoap.org/ws/2004/09/policy/optimizedmimeserializat
3567 ion" xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
3568 wssecurity-secext-1.0.xsd"
3569 xmlns:fimac="urn:de:egov:names:fim:1.0:authenticationcontext"
3570 xmlns:osci="http://www.osci.eu/ws/2008/05/transport.xsd"
3571 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
3572   <wsp:All>
3573
3574     <wsp:Policy wsu:id="X509CertificateAssertion">
3575       <osci:X509CertificateAssertion>
3576         <wsp:ALL>
3577           <wsse:SecurityTokenReference wsu:id="UltimateRecipientEncCert"
3578 wsse:Usage="http://www.osci.eu/2008/05/common/names/TokenUsage/e2eContentEncryption
3579 " osci:Role="http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver">
3580             <wsse:Reference URI="REPLACE_WITH_ACTUAL_URL to UltimateRecipientEncCert"
3581 ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-
3582 profile-1.0#X509v3"/>
3583           </wsse:SecurityTokenReference>
3584           <wsse:SecurityTokenReference wsu:id="RecipientSigCert"
3585 wsse:Usage="http://www.osci.eu/2008/05/common/names/TokenUsage/ReceiptSigning"
3586 osci:Role="http://www.osci.eu/ws/2008/05/transport/role/Recipient">
3587             <wsse:Reference URI="REPLACE_WITH_ACTUAL_URL to RecipientSigCert"
3588 ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-
3589 profile-1.0#X509v3"/>
3590           </wsse:SecurityTokenReference>
3591           <wsse:SecurityTokenReference wsu:id="MsgBoxEncCert"
3592 wsse:Usage="http://www.osci.eu/2008/05/common/names/TokenUsage/TransportEncryption"
3593 osci:Role="http://www.osci.eu/2008/05/common/names/role/MsgBox">
3594             <wsse:Reference URI="REPLACE_WITH_ACTUAL_URL to MsgBoxEncCert"
3595 ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-
3596 profile-1.0#X509v3"/>
3597           </wsse:SecurityTokenReference>
3598           <wsse:SecurityTokenReference wsu:id="MsgBoxSigCert"
3599 wsse:Usage="http://www.osci.eu/2008/05/common/names/TokenUsage/ReceiptSigning"
3600 osci:Role="http://www.osci.eu/2008/05/common/names/role/MsgBox">
3601             <wsse:Reference URI="REPLACE_WITH_ACTUAL_URL to MsgBoxSigCert"
3602 ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-
3603 profile-1.0#X509v3"/>
3604           </wsse:SecurityTokenReference>
3605         </wsp:ALL>
3606       </osci:X509CertificateAssertion>
3607     </wsp:Policy>
3608
3609     <wsp:Policy wsu:id="ServicesAssertion">
3610       <osci:ObsoleteAfterAssertion
3611 PolyRef="http://www.OSCIExampleEndPoint/MsgRetainPolicy.htm">

```

```
3612     <osci:MsgRetainDays>7</osci:MsgRetainDays>
3613     </osci:ObsoleteAfterAssertion>
3614 </wsp:Policy>
3615
3616     <wsp:Policy wsu:id="AuthLevelPolicy">
3617
3618     <fimac:Authentication>urn:de:egov:names:fim:1.0:securitylevel:high</fimac:Authenti
3619 cation>
3620
3621     <fimac:Registration>urn:de:egov:names:fim:1.0:securitylevel:veryhigh</fimac:Regist
3622 ration>
3623     </wsp:Policy>
3624 </wsp:All>
3625 </wsp:Policy>
```

3626 Listing 1: ExampleEndpointOSCIPolicy.xml

Appendix C. Example Signature Element

3627

3628

For illustration, following example is given for an instance of such a signature element:

3629

```
<ds:Signature Id="uuid:E57C0006-A629-5767-ED32-2667F1512912">
```

3630

```
  <ds:SignedInfo>
```

3631

```
    <ds:CanonicalizationMethod Algorithm=
```

3632

```
      "http://www.w3.org/2001/10/xml-exc-c14n#" />
```

3633

```
      <ds:SignatureMethod Algorithm=
```

3634

```
        "http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/>
```

3635

```
      <ds:Reference URI="#uuid:97544A28-F042-9457-3286-DD37F6FF7FEA">
```

3636

```
        <ds:Transforms>
```

3637

```
          <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
```

3638

```
        </ds:Transforms>
```

3639

```
          <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
```

3640

```
          <ds:DigestValue>DQrljZZVeewWoXLzLLi/uPqESY2fGscAjVLBXpjKEnM=</ds:DigestValue>
```

3641

```
        </ds:Reference>
```

3642

```
    <ds:Reference Id="uuid:4422AB49-BF3E-8521-BD1D-820F2160DDC6"
```

3643

```
      Type="http://uri.etsi.org/01903/v1.1.1/#SignedProperties"
```

3644

```
      URI="#uuid:5A075139-52E8-CF5E-3A1B-F54B6B1F1025">
```

3645

```
        <ds:Transforms>
```

3646

```
          <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
```

3647

```
        </ds:Transforms>
```

3648

```
          <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
```

3649

```
          <ds:DigestValue >RjwjB12TBumKSvG05Jgelzz6jllilA3t7GUxikLaa8io=</ds:DigestValue>
```

3650

```
        </ds:Reference>
```

3651

```
    </ds:SignedInfo>
```

3652

```
    <ds:SignatureValue>FrPlHt0v/Njnk...8JZV/LE141aStcLyBxBQ==</ds:SignatureValue>
```

3653

```
    <ds:KeyInfo >
```

3654

```
      <ds:X509Data >
```

3655

```
        <ds:X509Certificate>MIIDHjCCAgagAwIBAAIER4...YQya8Q==</ds:X509Certificate>
```

3656

```
      </ds:X509Data>
```

3657

```
    </ds:KeyInfo>
```

3658

```
    <ds:Object>
```

3659

```
      <xades:QualifyingProperties Target="#uuid:E57C0006-A629-5767-ED32-2667F1512912">
```

3660

```
        <xades:SignedProperties>
```

3661

```
          <xades:SignedSignatureProperties
```

3662

```
            Id="uuid:5A075139-52E8-CF5E-3A1B-F54B6B1F1025">
```

3663

```
              <xades:SigningTime>2008-01-17T18:57:27</xades:SigningTime>
```

3664

```
              <xades:SigningCertificate>
```

3665

```
                <xades:Cert>
```

3666

```
                  <xades:CertDigest>
```

3667

```
                    <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
```

3668

```
                    <ds:DigestValue >0uGTT8Gg...oXRjpsKp9BMVBaYid7kzsa4=</ds:DigestValue>
```

```
3669     </xades:CertDigest>
3670     <xades:IssuerSerial>
3671         <ds:X509IssuerName>CN=Apitzsch, OU=QA, O=waycony, L=Hamburg, C=DE
3672     </ds:X509IssuerName>
3673         <ds:X509SerialNumber >1200577645</ds:X509SerialNumber>
3674     </xades:IssuerSerial>
3675 </xades:Cert>
3676 </xades:SigningCertificate>
3677 </xades:SignedSignatureProperties>
3678 </xades:SignedProperties>
3679 <xades:UnsignedProperties>
3680     <xades:UnsignedSignatureProperties>^
3681         <xades:SignatureTimeStamp>
3682         <ds:CanonicalizationMethod Algorithm=
3683             "http://www.w3.org/2001/10/xml-exc-c14n#" />
3684         <xades:EncapsulatedTimeStamp>0uGKT8Gg..oXRjpsKp9BMVBaYid
3685     </xades:EncapsulatedTimeStamp>
3686     </xades:SignatureTimeStamp>
3687 </xades:UnsignedSignatureProperties>
3688 </xades:UnsignedProperties>
3689 </xades:QualifyingProperties>
3690 </ds:Object>
3691 </ds:Signature>
```

3692 Listing 2: Example XML Signature

3693 **Appendix D. Acknowledgements**

3694 Following people having contributed temporarily or during the whole specification process to the
3695 OSCI Transport 2 concepts and documents are gratefully acknowledged:

3696 **Requirements, Architecture and Specification Working Groups**

3697 Jörg Apitzsch (bos), Ingo Beyer (PC-Ware), Thomas Biere (BSI), Oliver Böhm (Fraunhofer ISST), Nils
3698 Büngener (bos), Dr. Peter Dettling (IBM Deutschland), Jan Füssel (cit), Clemens Gogolin (PTB), Golo
3699 Hoffmann (procilon), Marc Horstmann (bos), Christoph Karich (Hochschule Harz), Daniel Koszior
3700 (PC-Ware), Harald Krause (Dataport), Arnold Külper (DVZ Mecklenburg-Vorpommern), Raik Kuhlisch
3701 (Fraunhofer ISST), Ralf Lindemann (bos), Dr. Klaus Lüttich (bos), Fabian Meiswinkel (Microsoft
3702 Deutschland), Lutz Nentwig (Fraunhofer ISST), Lars Nitzsche (procilon), Torsten Rienaß (procilon),
3703 Martin Schacht (Microsoft Deutschland), Thilo Schuster (cit), Janos Schwellach (bos), Prof. Dr.-Ing.
3704 Hermann Strack (Hochschule Harz), Dr. Hamed Tabrizi (bos), Lutz Vorwerk (IZN Niedersachsen),
3705 Sascha Weinreuter (cit), Mario Wendt (Microsoft Deutschland)

3706 **Approval Instance**

3707 Members of the Architecture and Specification Working Group and:

3708 Marcel Boffo (LDI Rheinland-Pfalz), Carlheinz Braun (DPMA), Christoph Damm (Staatskanzlei
3709 Sachsen), Steffen Düring (UBA), Joachim Gerber (INFORA), Reto Giger (Schweizer Post), Jens
3710 Habermann (LDS Düsseldorf), Renée Hinz (UBA), Wolfgang Klebsattel (DLR), Andreas Kraft (PBEG),
3711 Svea Lahn (HSH), Dr. Christian Mrugalla (BMI), Dr. Bernhard Paul (IBM Deutschland), Maren Pohl
3712 (HABIT), Anja Riekenberg (Hannit), Martin Rost (ULD Kiel), Alexander Spohn (ITDZ), Frank Steimke
3713 (OSCI Leitstelle), Andrea Steinbeck (HSH), Heiko Thede (DVZ Mecklenburg-Vorpommern), Joachim
3714 Wille (SAP Deutschland)