



Koordinierungsstelle  
für IT-Standards



1

# OSCI-Transport, Version 2.0.2

2

– Web Services Profiling and Extensions Specification –

3

**Koordinierungsstelle für IT-Standards (KoSIT)**

4

**(Standardisation Office of the Federal German Government)**

5

**Version 2.0.2**

6

***Final***

7

Last edited March 13, 2015

8

9 This is a proposal for the revisions of the approved final version of the OSCI Transport Version 2.0.1  
10 Web Services Profiling and Extensions Specification. Minor clarifications may be eligible, which could  
11 result from perceptions made in the implementation and/or rollout process. These will be published in  
12 future editions of this document. An overview of changes made to initial the version 2.0.1 of this  
13 specification is provided in Appendix D.

14 Reason for this revision: Changes applied to section 8.4, MessageMetaData

- 15 • adoption of xoev:Codelist type for some elements;
  - 16 • eliminating QName typed attributes/elements;
  - 17 • PartyType elements now may include optional SecurityTokens (as e.g. used in XVergabe)
- 18 and
- 19 • where applicable, references to specifications updated to most recent versions.

20 The latest edition and schema files will always be available at: <http://www.xoev.de/de/download>.

21

22 Editor of this document:

23 Jörg Apitzsch,

24 Governikus GmbH & Co. KG,

25 E-Mail: [joerg.apitzsch@governikus.com](mailto:joerg.apitzsch@governikus.com)

26 Quality Assurance:

27 Ulrich Horst, Ralf Lindemann

28 Governikus GmbH & Co. KG,

29

30 Further contributors are listed in Appendix E.

31

32 Comments and questions may be addressed to:

33 Die Senatorin für Finanzen

34 – 02 – Zentrales IT-Management und E-Government

35 Koordinierungsstelle für IT-Standards (KoSIT)

36

37 Rudolf-Hilferding-Platz 1

38 28195 Bremen, Germany

39 Tel.: +49 421 361 19739

40 E-Mail: [kosit@finanzen.bremen.de](mailto:kosit@finanzen.bremen.de)

41 or to the editor directly.

42

## Table of Contents

43	1	Introduction.....	6
44	2	Document Structure .....	7
45	3	Document Conventions .....	8
46	3.1	Notational Conventions .....	8
47	3.2	XML Namespaces.....	10
48	4	Specification Conformance .....	12
49	4.1	Conformance Requirements .....	12
50	4.2	OSCI Roles and Conformance Targets .....	12
51	5	SOAP Version, Transport and Fault Binding.....	14
52	5.1	General processing error .....	14
53	5.2	Fault Delivery, Logging and Escalation.....	15
54	6	Addressing Endpoints .....	16
55	6.1	Use of WS-Addressing.....	16
56	6.1.1	Endpoint Reference .....	16
57	6.1.2	Addressing Properties – SOAP Binding .....	18
58	6.2	Non addressable Initiators and use of WS MakeConnection .....	20
59	6.3	Addressing faults.....	21
60	7	Message Security, Authentication and Authorization.....	22
61	7.1	WS Security Header Block.....	22
62	7.2	XML Digital Signature .....	22
63	7.2.1	Restrictions to WS-I Basic Security Profiling .....	22
64	7.2.2	Format of XML Digital Signatures used for Documents .....	23
65	7.3	XML Encryption.....	26
66	7.3.1	End-to-end Encryption of Content Data.....	26
67	7.3.2	Encryption Cyphersuite Restrictions.....	27
68	7.4	Security Token Types .....	27
69	7.5	Use of WS-Trust and SAML Token.....	28
70	7.5.1	Authentication Strongness.....	29
71	7.5.2	WS-Trust Messages .....	30
72	7.5.3	Issued SAML-Token Details .....	36
73	7.5.4	Authentication for Foreign Domain Access .....	38
74	7.5.5	SAML-Token for Receipt/Notification Delivery .....	38
75	8	OSCI Specific Extensions .....	42
76	8.1	Message Flow Time Stamping.....	42
77	8.2	Accessing Message Boxes .....	43

78	8.2.1	MsgBoxFetchRequest .....	43
79	8.2.2	MsgBoxStatusListRequest.....	46
80	8.2.3	MsgBoxResponse.....	48
81	8.2.4	MsgBoxGetNextRequest .....	52
82	8.2.5	MsgBoxCloseRequest .....	53
83	8.2.6	Processing Rules for MsgBoxGetNext/CloseRequest.....	55
84	8.3	Receipts .....	55
85	8.3.1	Demanding Receipts .....	56
86	8.3.2	Receipt Format and Processing .....	59
87	8.3.3	Submission and Relay Receipt.....	64
88	8.3.4	Fetches Notification .....	64
89	8.3.5	Additional Receipt/Notification Demand Fault Processing Rules .....	66
90	8.4	Message Meta Data .....	67
91	8.4.1	Re-used Type Definitions .....	67
92	8.4.2	Description of Message Meta Data Header.....	72
93	8.5	X.509-Token Validation on the Message Route .....	79
94	8.5.1	X.509-Token Container.....	80
95	8.5.2	X.509-Token Validation Results .....	82
96	8.5.3	Verification of XKMS Validate Result Signatures .....	82
97	8.6	General Processing of Custom Header Faults .....	83
98	9	Constituents of OSCI Message Types .....	84
99	9.1	osci:Request .....	85
100	9.2	osci:Response.....	87
101	9.3	MsgBoxFetchRequest.....	89
102	9.4	MsgBoxStatusListRequest .....	90
103	9.5	MsgBoxResponse .....	91
104	9.6	MsgBoxGetNextRequest.....	92
105	9.7	MsgBoxCloseRequest.....	94
106	10	Policies and Metadata of Communication Nodes and Endpoints.....	96
107	10.1	General Usage of Web Service Description Language .....	96
108	10.1.1	WSDL and Policies for MEP Synchronous Point-To-Point.....	96
109	10.1.2	WSDL and Policies for Asynchronous MEPs via Message Boxes.....	97
110	10.2	OSCI Specific Characteristics of Endpoints .....	97
111	10.2.1	Certificates used for Signatures and Encryption .....	97
112	10.2.2	Endpoint Services and Limitations .....	101
113	10.3	WS Addressing Metadata and WS MakeConnection .....	103
114	10.4	WS Reliable Messaging Policy Assertions .....	104

115	10.5	MTOM Policy Assertion .....	104
116	10.6	WS Security Profile and Policy Assertions .....	105
117	10.6.1	Endpoint Policy Subject Assertions .....	105
118	10.6.2	Message Policy Subject Assertions.....	106
119	10.6.3	Algorithm Suite Assertions.....	107
120	11	Applying End-to-end Encryption and Digital Signatures on Content Data .....	108
121	12	Indices.....	109
122	12.1	Tables .....	109
123	12.2	Pictures .....	109
124	12.3	OSCI specific faults .....	109
125	12.4	Listings.....	110
126	13	References.....	111
127	13.1	Normative.....	111
128	13.2	Informative .....	114
129		Appendix A. Schema OSCI Transport 2.02.....	115
130		Appendix B. OSCI Transport 2.02 – Schema MessageMetaData .....	121
131		Appendix C. Example: OSCI Endpoint Metadata Instance .....	121
132		Appendix D. Change History .....	130
133		Appendix E. Acknowledgements.....	131

## 134 **1 Introduction**

135 This version 2.0.2 of the Web Services Profiling and Extensions Specification **Online Services**  
136 **Computer Interface Transport (OSCI)** replaces the former version 2.0.1. The initial version 2.0 was  
137 made up of four documents:

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

139 (2) "OSCI-Transport 2 – Features and Architecture Overview"

140 and

141 (3) "OSCI Transport 2.0 – Web Services Profiling and Extensions Specification".

142 These documents are accomplished by a common comprehensive glossary:

143 (4) "OSCI Transport 2 – Glossary".

144 While the technical overview and the specification and profiling documents are presented in English  
145 language only, the other mentioned documents are available in German language.

146 The background and principles of the **Online Service Computer Interface (OSCI) Transport**  
147 specification is explained in the document "OSCI-Transport 2 – Features and Architecture Overview",  
148 which should be read first to obtain a base understanding for the profiling and specifications outlined in  
149 the here presented document.

150

151 Main motivation for this update is to deal with new requirements recognized in OSCI Transport  
152 application scenarios in the time span since the first publication of version 2.0 in 2007:

- 153 • slight revision of the OSCI role model, according the one defined by former OSCI Transport  
154 version 1.2
- 155 • possibility to carry extended, flexible meta data about (opaque) message payload as well as  
156 transport related information
- 157 • use of logical identifiers for communication partners
- 158 • enhanced flexibility and extensibility concerning message types, SOAP faults, and on the  
159 recipient side, OSCI message box access
- 160 • new receipt types foreseen for message-submission and -relaying.

## 161 **2 Document Structure**

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

164 Due to the proliferation of differing platforms and technologies in the e-government, it is essential to  
165 ensure that the different web service implementations are interoperable, regardless of the underlying  
166 implementation and operation technology. Therefore, we mainly rely on the work, which is done by the  
167 Web Services Interoperability Organization<sup>1</sup>, where a profiling is compiled of the major web services  
168 specifications under the aspect of best practices for web services interoperability. If needed to satisfy  
169 the underlying OSCI requirements, we define further restrictions and processing rules in addition to  
170 this profiling. This is outlined in the chapters [5 through 7].

171 As OSCI Transport has to serve special requirements, which are not yet satisfied by currently  
172 available web services specifications, chapter [8] specifies extensions to the WS-Stack for these  
173 purposes.

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

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

---

<sup>1</sup> see <http://www.ws-i.org/>

## 181 3 Document Conventions

### 182 3.1 Notational Conventions

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

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

- 187 • The syntax appears as an XML instance, but values in italics indicate data types instead of  
188 values.
- 189 • Characters are appended to elements and attributes to indicate cardinality:
  - 190 ○ "?" (0 or 1)
  - 191 ○ "\*" (0 or more)
  - 192 ○ "+" (1 or more)
- 193 • The character "|" is used to indicate a choice between alternatives.
- 194 • The characters "(" and ")" are used to indicate that contained items are to be treated as a  
195 group with respect to cardinality or choice.
- 196 • An ellipsis (i.e. "...") indicates a point of extensibility that allows other child or attributes content  
197 specified in this document. Additional children elements and/or attributes MAY be added at the  
198 indicated extension points but they MUST NOT contradict the semantics of the parent and/or  
199 owner, respectively. If an extension is not recognized it SHOULD be ignored.
- 200 • XML namespace prefixes (see section 3.2) are used to indicate the namespace of the element  
201 being defined.

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

- 204 • An element extensibility point is referred to using {any} in place of the element name. This  
205 indicates that any element name can be used, from any namespace other than the osci: or  
206 oscimeta: namespaces.
- 207 • An attribute extensibility point is referred to using @{any} in place of the attribute name. This  
208 indicates that any attribute name from any namespace can be used.

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

211 **Rnnnn** - *Statement text here*

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

214 The terms "header" and "body" used in this document are used as abbreviation of "SOAP header" and  
215 "SOAP body" respectively.

216 The following legend applies to the message diagrams in this document:

- 217 • Mandatory constituents have continuous lines, optional ones are marked dashed.
- 218 • If present, arrows on the left diagram side mark transport encryption requirements, those on  
219 the right transport signature requirements. If not present, general according requirements are  
220 described in textual form.
- 221 • Encrypted message parts are marked by a hatched background.
- 222 • All arrows illustrate transport encryption and signature based on WS Security asymmetrical  
223 binding; if symmetrical binding is used, encryption and signature is applied using HTTPS.

224 For an explanation of used abbreviations and terms see the additional document "OSCI Transport 2.0  
225 – Glossary".

226 **Note:** For ease of identification, important changes and extensions introduced with this specification  
227 version related to the former 2.0 version are highlighted by a turquoise background.

228

229 **3.2 XML Namespaces**

230 The following XML namespaces are referenced:

Prefix	XML Namespace	Specification
ds	<a href="http://www.w3.org/2000/09/xmldsig#">http://www.w3.org/2000/09/xmldsig#</a>	[XMLDSIG]
dss	<a href="urn:oasis:names:tc:dss:1.0:core:schema">urn:oasis:names:tc:dss:1.0:core:schema</a>	[DSS][DSS]
fimac	<a href="urn:de:egov:names:fim:1.0:authenticationcontext&lt;sup&gt;2&lt;/sup">urn:de:egov:names:fim:1.0:authenticationcontext<sup>2</sup></a>	[SAFE]
osci	<a href="http://www.osci.eu/ws/2008/05/transport">http://www.osci.eu/ws/2008/05/transport</a>	This document
oscimeta	<a href="http://www.osci.eu/ws/2014/10/transport">http://www.osci.eu/ws/2014/10/transport</a>	This document
s12	<a href="http://www.w3.org/2003/05/soap-envelope">http://www.w3.org/2003/05/soap-envelope</a>	[SOAP12]
samlac	<a href="urn:oasis:names:tc:SAML:2.0:ac">urn:oasis:names:tc:SAML:2.0:ac</a>	[SAMLAC]
saml1	<a href="urn:oasis:names:tc:SAML:1.0:assertion">urn:oasis:names:tc:SAML:1.0:assertion</a>	[SAML1]
saml2	<a href="urn:oasis:names:tc:SAML:2.0:assertion">urn:oasis:names:tc:SAML:2.0:assertion</a>	[SAML2]
wsa	<a href="http://www.w3.org/2005/08/addressing">http://www.w3.org/2005/08/addressing</a>	[WSA]
wsaw	<a href="http://www.w3.org/2006/05/addressing/wsdl">http://www.w3.org/2006/05/addressing/wsdl</a>	[WSAW]
wsdl	<a href="http://www.w3.org/ns/wsdl-instance">http://www.w3.org/ns/wsdl-instance</a>	[WSDL20]
wsdl11	<a href="http://schemas.xmlsoap.org/wsdl/">http://schemas.xmlsoap.org/wsdl/</a>	[WSDL11][WSDL11]
wsmc	<a href="http://docs.oasis-open.org/ws-rx/wsmc/200702">http://docs.oasis-open.org/ws-rx/wsmc/200702</a>	[WSMC]
wsp	<a href="http://www.w3.org/ns/ws-policy">http://www.w3.org/ns/ws-policy</a>	[WSPF], [WSPA]
wspmtom	<a href="http://docs.oasis-open.org/ws-rx/wsrmp/200702">http://docs.oasis-open.org/ws-rx/wsrmp/200702</a>	[MTOMP]
wstrm	<a href="http://docs.oasis-open.org/ws-rx/wstrm/200702">http://docs.oasis-open.org/ws-rx/wstrm/200702</a>	[WSRM]
wstrmp	<a href="http://docs.oasis-open.org/ws-rx/wstrmp/200702">http://docs.oasis-open.org/ws-rx/wstrmp/200702</a>	[WSRMP]
wsse	<a href="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd</a>	[WSS]
wssp	<a href="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702">http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702</a>	[WSSP]
wst	<a href="http://docs.oasis-open.org/ws-sx/ws-trust/200512">http://docs.oasis-open.org/ws-sx/ws-trust/200512</a>	[WST]
wsu	<a href="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd</a>	[WSS]
xenc	<a href="http://www.w3.org/2001/04/xmlenc#">http://www.w3.org/2001/04/xmlenc#</a>	[XENC]

<sup>2</sup> Preliminary namespace for a SAML AuthnContext extension; proposal subject to standardization in Germany

xkms	<a href="http://www.w3.org/2002/03/xkms#">http://www.w3.org/2002/03/xkms#</a>	[XKMS]
xkmsEU	<a href="http://uri.peppol.eu/xkmsExt/v2#">http://uri.peppol.eu/xkmsExt/v2#</a>	[XKMSEU]
xoev-dt	<a href="http://xoev.de/schemata/basisdatentypen/1_1">http://xoev.de/schemata/basisdatentypen/1_1</a>	[XOEVHB]
xs	<a href="http://www.w3.org/2001/XMLSchema">http://www.w3.org/2001/XMLSchema</a>	[XMLSchema]

231 Table 1: Referenced Namespaces

## 232 4 Specification Conformance

### 233 4.1 Conformance Requirements

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

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

- 239 • <http://www.osci.eu/ws/2008/05/osci-transport>
- 240 • <http://www.osci.eu/ws/2014/10/osci-transport>
- 241 • [http://xoev.de/schemata/basisdatentypen/1\\_1](http://xoev.de/schemata/basisdatentypen/1_1)
- 242 • <http://www.w3.org/2002/03/xkms#>
- 243 • <http://uri.peppol.eu/xkmsExt/v2#>.

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

### 246 4.2 OSCI Roles and Conformance Targets

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

249 This allows for the definition of conformance in different contexts, to assure unambiguous  
250 interpretation of the applicability of requirements, and to allow conformance testing of artefacts (e.g.,  
251 SOAP messages and WSDL descriptions) and the behaviour of various parties to a web service (e.g.,  
252 clients and service instances).

253 Requirements conformance targets are physical artefacts wherever possible, to simplify testing and  
254 avoid ambiguity.

255 The following conformance targets are used in this specification (for target names, synonyms are  
256 mentioned, if often used in referenced web service specifications):

257 **OSCI MESSAGE** - protocol elements that profile the SOAP envelope, whereby following special OSCI  
258 message types are defined:

259 **osci:Request, osci:Response, MsgBoxFetchRequest, MsgBoxResponse,**  
260 **MsgBoxStatusListRequest, MsgBoxGetNextRequest, MsgBoxCloseRequest**

261 **PAYLOAD** (OSCI synonym: **Content Data**) – message payload, produced by AUTHOR, designated  
262 to be consumed by READER; the transport infrastructure is agnostic about structure and content of  
263 **PAYLOAD**.

264 **OSCI GATEWAY** – an assembly of functionalities realized in software, able to produce, send, receive,  
265 and consume OSCI messages, hereby not concerned with SOAP body entries (OSCI messages for  
266 MsgBox access and faults transmitted in the SOAP body excepted)

267 **DESCRIPTION** - descriptions of types, messages, interfaces and their concrete protocol and data  
268 format bindings, and the network access points associated with web services (e.g., WSDL  
269 descriptions)

270 **AUTHOR** (synonym: **Requester, Source Application**) – end point instance that wishes to use a  
271 **PROVIDER** entity (OSCI-term: READER) web service, providing according message **PAYLOAD** and  
272 initial message transport attributes. An author uses an **INITIATOR** instance for dispatching messages.

273 **READER** (synonym: **(Service) Provider, Target Application**) – end point instance providing services  
 274 for AUTHORS, acting on the request message PAYLOAD and expected to produce related response  
 275 messages. A READER uses a RECIPIENT instance for receiving messages.

276 **INITIATOR** (synonym: **Sender**) – software agent used by the AUTHOR that generates a message  
 277 according to the protocol associated with it and that transmits it to a RECIPIENT or MsgBox,  
 278 potentially through a message path that involves one or multiple INTERMEDIARY(ies)

279 **RECIPIENT** – software agent that receives a message according to the protocol associated with it.

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

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

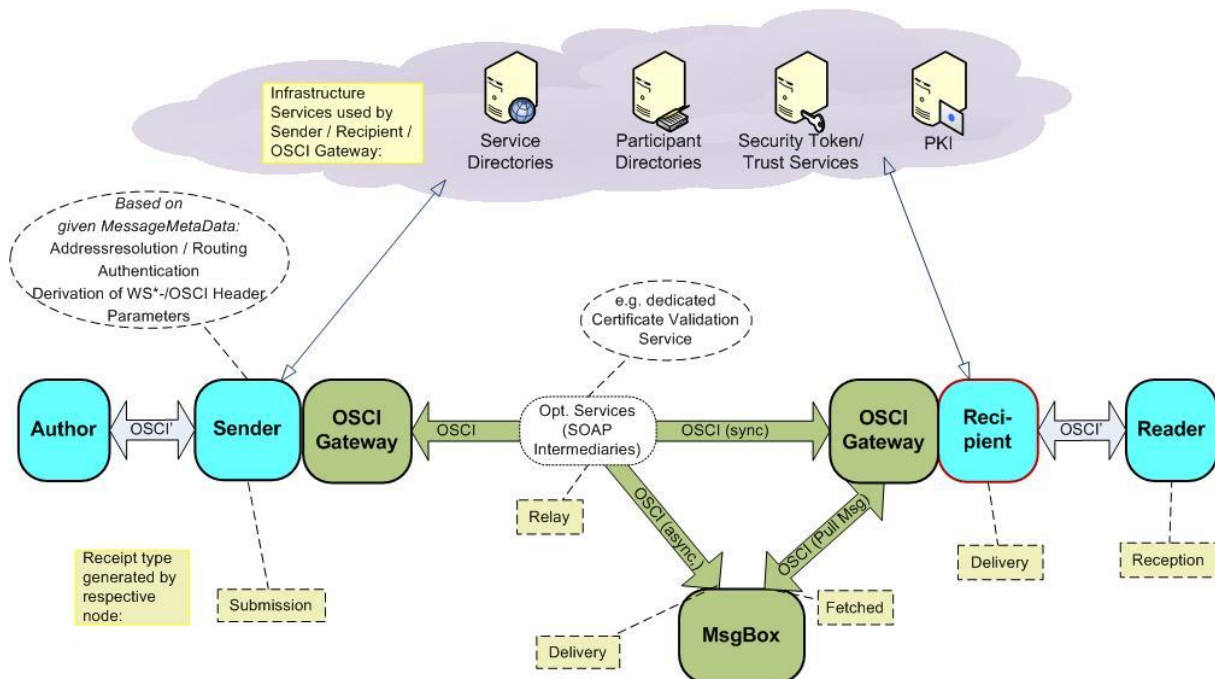
284 **ENDPOINT** – collective term for INITIATOR, RECIPIENT, and MsgBox. Each ENDPOINT may be in  
 285 the role of a Security Token Requestor (STR)

286 **STR** – Security Token Requestor as defined by WS-Trust.

287 **STS** – Security Token Service as defined by WS-Trust.

288 **SAML-TOKEN** – Security Token as defined by SAML.

289 The following picture gives a brief overview of the message flow and respective roles of involved  
 290 nodes:



291

292 Figure 1: Actors and nodes involved in the message flow

293 Olive marked nodes and message routes were addressed by version 2.0 of OSCI Transport, in  
 294 version 2.01 now extended by nodes marked turquoise. The message routes named OSCI do not  
 295 need the full set of WS-\* SOAP headers, these are mostly based on the header  
 296 `oscimeta:MessageMetaData` described in chapter [8.4].

297 **Note:** For the realisation of the OSCI route, implementations may choose a network connection based  
 298 on SOAP, a local API interface, or e.g. products offering SOA bus means.

## 299 5 SOAP Version, Transport and Fault Binding

300 **R0010** - **OSCI Nodes** MUST support SOAP Version 1.2 according to [SOAP12] and constraints  
301 specified in [WSI-Basic], chapter 3 Messaging with restriction **R0020**.

302 For ease of implementation and interoperability of web services involved, for the payload carried in the  
303 SOAP body it is **STRONGLY RECOMMENDED** to include only one child element, using the  
304 document-literal binding as defined by the Web Service Description Language [WSDL11].

305 **R0020** - Transport binding is restricted to HTTP/1.1, which has performance advantages and is  
306 more clearly specified than HTTP/1.0. R1140 of [WSI-Basic] (A MESSAGE SHOULD be  
307 sent using HTTP/1.1) – is superseded: A **MESSAGE** MUST be sent using HTTP/1.1.

308 Note that this requirement does not prohibit the use of HTTPS.

309 **R0030** - Errors use the SOAP fault mechanisms. The SOAP fault block according to [SOAP12]  
310 MUST be used to report information about errors occurring while processing a  
311 SOAP/OSCI message. The `s12:Fault` element MUST be carried in the SOAP body  
312 block of the network backchannel SOAP response message or – if no backchannel is  
313 available in asynchronous scenarios – in the SOAP body block of a distinct message of  
314 `osci:Request`.

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

317 Implementations may have the need to additionally define SOAP faults according to their needs. To  
318 assure awareness of those faults by all implementations of this specification, they MUST be brought  
319 according to fault situation signalling and message delivery interruption by nodes receiving such a  
320 SOAP fault.

321 The following information for the subelements `s12:Fault` is supplied per fault described in this  
322 document:

323 Sub Element	Property Label	Possible Values
324 <code>/Fault/Code/Value</code>	[Code]	Sender   Receiver
325 <code>/Fault/Code/Value/Subcode</code>	[Subcode]	A local QName assigned to the fault
326 <code>/Fault/Reason/Text</code>	[Reason]	The English language reason explanation

327 In the fault message itself, **faults defined in this specification** the [Code] value MUST have a prefix of  
328 `s12:;` the [Subcode] value prefix MUST be `osci:.`

329 It should be noted that implementations MAY provide second-level details fields, but they should be  
330 careful not to introduce security vulnerabilities when doing so (e.g. by providing too detailed  
331 information).

### 332 5.1 General processing error

333 If an unspecific and unrecoverable message processing error occurs, a fault MUST be generated and  
334 the message MUST be discarded.

335 **NOTE:** There MUST NOT be generated a [Subcode] value prefix in this case!

336	Fault 1: <b>ProcessingException</b>	
337	[Code]	Receiver
338	[Subcode]	ProcessingException
339	[Reason]	Unspecific processing error

340 Implementations MAY provide second-level details fields, e.g. a stack trace, if this information does  
341 not lead to security vulnerabilities (see advise above).

## 342 **5.2 Fault Delivery, Logging and Escalation**

343 In general, the fault handling defined in [SOAP12], chapter 5.4 "SOAP Fault" applies as well as the  
344 respective fault handlings defined by the OSCI incorporated specifications. Normally faults should be  
345 raised in situations where the initiator can be informed directly about this fact. The fault MUST be  
346 logged by the node where the fault raises to be available for supervision and revision purposes. If  
347 faults arise at the node a message is targeted to, an according SOAP fault MUST be delivered in  
348 HTTP backchannel of the underlying request. Message processing MUST be aborted, if not specified  
349 otherwise for special situations in this document.

350 Though, situations exist where the possibility to deliver this information to the initiating node of the  
351 underlying message does not exist. In this case, appropriate escalation mechanisms MUST be  
352 foreseen by conformant implementations to signal such situations to the system monitoring  
353 environment / operating personal; follow-up of this situation is up to the operating policies<sup>3</sup>.

---

<sup>3</sup> Those should be made available online for all possible communication partners. Details are not addressed by this document.

## 354 6 Addressing Endpoints

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

357 **R0100** - **OSCI Nodes** MUST support WS-Addressing and WS-Addressing Metadata according to  
358 [WSA] and [WSAM]. Constraints apply specified in [WSI-Basic], chapter 3.7 " WS-  
359 Addressing Support".

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

### 362 6.1 Use of WS-Addressing

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

#### 366 6.1.1 Endpoint Reference

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

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

```

372 <wsa:EndpointReference>
373 <wsa:Address> xs:anyURI </wsa:Address>
374 <wsa:ReferenceParameters>
375 <osci:TypeOfBusinessScenario>
376   osci:TypeofBusinessScenarioType
377 </osci:TypeOfBusinessScenario>
378 </wsa:ReferenceParameters> ?
379 <wsa:Metadata>
380   ( xmlns:w3org="http://www.w3.org/ns/w3org" |
381     w3org:w3orgLocation="xs:anyURI xs:anyURI" > ) |
382   xs:any*
383 </wsa:Metadata> ?
384 </wsa:EndpointReference>
385 <documentation> type definition of osci:TypeOfBusinessScenario
386 </documentation>
387 <osci:TypeOfBusinessScenarioType>
388 <xs:simpleContent>
389   <xs:extension base="xs:anyURI">
390     <xs:attribute ref="wsa:IsReferenceParameters" use="optional"/>
391   </xs:extension>
392 </xs:simpleContent>
393 </osci:TypeOfBusinessScenarioType>

```

394 Description of outline above:

395 **/wsa:ReferenceParameters**

396 **R0120** – If the URI value of `.../wsa:Address` is not equal to  
397 `"http://www.w3.org/2005/08/addressing/anonymous"`, an EPR MUST contain  
398 one element **wsa:ReferenceParameters** which carries the type of business scenario  
399 addressed by the message. This element is defined as type `xs:any*` and optional in [WSA].  
400 The type of business scenario MUST be tagged as URI in the OSCI namespace as  
401 **osci:TypeOfBusinessScenario**.(see below).

402 Any endpoint SHOULD expose the types of business scenarios which it actually is able to  
403 serve in WSDL [WSDL11] format. An XML schema definition for the Content Data to be

404 carried in the SOAP body of the message MUST be bound to the concrete tagged type of  
 405 business scenario. Following the WSDL binding of WS-Addressing [WSAW], each  
 406 `osci:TypeOfBusinessScenario` corresponds to a specific port [WSDL11] respective endpoint  
 407 [WSDL20].

408 `/osci:TypeOfBusinessScenario`

409 The type of business scenario MUST be outlined as URI in the OSCI namespace.

410 `/osci:TypeOfBusinessScenario/@wsa:IsReferenceParameter`

411 Attribute of type `xs:boolean` as described in [WSA] . Value is always true.

412 The following types of business scenarios MUST be served by all OSCI endpoints:

Type of business scenario URI	Meaning
<code>http://www.osci.eu/ws/2008/05/common/urn/messageTypes/Receipt</code>	Receipt type messages
<code>http://www.osci.eu/ws/2008/05/common/urn/messageTypes/Notification</code>	Notification type messages
<code>http://www.osci.eu/ws/2008/05/common/urn/messageTypes/Fault</code>	Fault type messages

413 Table 2: Predefined business scenario types

414 The following type of business scenario SHOULD be served by OSCI endpoints, which are  
 415 intended to be able to support a common mail-style data exchange, optional carrying any  
 416 type of attachments<sup>4</sup>:

417 `http://www.osci.eu/ws/2008/05/common/urn/messageTypes/LetterStyle`

418 `/wsa:MetaData`

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

421 Each OSCI endpoint SHOULD publish a link to its WSDL by using `wsdl:wsdlLocation`.  
 422 Such elements define the metadata that is relevant to the interaction with the endpoint. An  
 423 initiator MUST have knowledge about the following metadata specific for OSCI about the  
 424 destination he is targeting a message to. At least, each OSCI endpoint SHOULD publish  
 425 references to its encryption and signature certificate(s) in the OSCI specific policy  
 426 `/osci:X509CertificateAssertion`<sup>5</sup> by using the  
 427 `wsse:SecurityTokenReference/wsse:Reference` token reference.

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

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

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

434 Availability of qualified time stamping service and policies, those apply here (as exposed in  
 435 `/osci:QualTSPAssertion`<sup>6</sup>).

<sup>4</sup> The according content data schemes are published together with its specification at  
<http://www.xoev.de/sixcms/detail.php?gsid=bremen83.c.2316.de#Standards>

<sup>5</sup> Details are defined in chapter [10.2.1]

<sup>6</sup> See chapter [10.2.2]

436 Possible rules applied by the destination concerning message lifetime, if messages are  
 437 marked as valid for a restricted period of time (see chapter [8.1] for this issue; the endpoint  
 438 behaviour is outlined in the `/osci:ObsoleteAfterAssertion`<sup>6</sup> of the OSCI specific  
 439 policy.

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

## 445 6.1.2 Addressing Properties – SOAP Binding

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

```

449 <wsa: To> xs: anyURI </ wsa: To>
450 <wsa: From> wsa: Endpoi nt Ref er enceType </ wsa: From> ?
451 <wsa: Repl yTo> wsa: Endpoi nt Ref er enceType </ wsa: Repl yTo> ?
452 <wsa: Faul t To> wsa: Endpoi nt Ref er enceType </ wsa: Faul t To> ?
453 <wsa: Act i on>
454 xs: anyURI |
455 ht t p: // www. osci . eu/ ws/ 2008/ 05/ t r a n s p o r t / u r n/ messageTypes/ OSCI Request |
456 ht t p: // www. osci . eu/ ws/ 2008/ 05/ t r a n s p o r t / u r n/ messageTypes/ OSCI Response |
457 ht t p: // www. osci . eu/ ws/ 2008/ 05/ t r a n s p o r t / u r n/ messageTypes/ MsgBoxFet chRequest
458 |
459 ht t p: // www. osci . eu/ ws/ 2008/ 05/ t r a n s p o r t / u r n/ messageTypes/ MsgBoxSt at usLi st Re
460 quest |
461 ht t p: // www. osci . eu/ ws/ 2008/ 05/ t r a n s p o r t / u r n/ messageTypes/ MsgBoxResponse |
462 ht t p: // www. osci . eu/ ws/ 2008/ 05/ t r a n s p o r t / u r n/ messageTypes/ MsgBoxGet Next Reque
463 st |
464 ht t p: // www. osci . eu/ ws/ 2008/ 05/ t r a n s p o r t / u r n/ messageTypes/ MsgBoxCl oseRequest
465 </ wsa: Act i on>
466 <wsa: Messagel D> xs: anyURI </ wsa: Messagel D>
467 <wsa: Rel at esTo Rel at i onshi pType=" xs: anyURI " ?>xs: anyURI </ wsa: Rel at esTo> *
468 <wsa: Ref er encePar amet er s>xs: any* </ wsa: Ref er encePar amet er s>
  
```

469 Description of outline above:

### 470 `/wsa:To`

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

### 473 `/wsa:From ?`

474 As OSCI is designed for authoritative communication, an OSCI message SHOULD carry at  
 475 most one SOAP header element `wsa:From` of type `wsa:EndpointReferenceType`. If  
 476 carried, the issuer of this message MUST expose here the EPR where he is able to accept  
 477 `osci:Request` messages according to R0120, R0130; it SHOULD carry the same entries as  
 478 `/wsa:ReplyTo`.

479 In case of an anonymous initiator this EPR MAY contain the only child element  
 480 `wsa:Address` with a content of  
 481 `"http://www.w3.org/2005/08/addressing/anonymous"`.

### 482 `/wsa:ReplyTo ?`

483 **R0140** - in case of non-anonymous and/or asynchronous scenarios messages of type  
 484 `osci:Request` MUST carry exactly one SOAP header element `wsa:ReplyTo` of type  
 485 `wsa:EndpointReferenceType`. This MUST contain the concrete EPR of the endpoint  
 486 according to R0120, R0130; it denotes the final destination to which the Recipient MUST  
 487 deliver the response. The `wsa:ReferenceParameters` of this EPR SHOULD be the  
 488 same as bound to the address element `wsa:To`.

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

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

494 **/wsa:FaultTo ?**

495 **R0150** - If faults related to this message shall not (or cannot in asynchronous scenarios) be  
 496 delivered in the network connection backchannel or it is intended to route such fault  
 497 messages to specialized endpoints for consuming fault messages, an OSCI message  
 498 SHOULD carry this optional element **wsa:FaultTo** of type  
 499 **wsa:EndpointReferenceType**. This MUST be a concrete EPR according to R1020,  
 500 R0130. To distinct such messages from other message types, the  
 501 **wsa:ReferenceParameters** of this EPR MUST be

```
502 <osci:TypeOfBusinessScenario>
503   http://www.osci.eu/ws/2008/05/common/urn/messageTypes/Fault
504 </osci:TypeOfBusinessScenario>
```

505 In this case, an HTTP response code 500 MUST be returned in the backchannel of the  
 506 SOAP request and the body of the SOAP response MUST carry the fault information in  
 507 parallel to the fault message sent to the endpoint denoted in **/wsa:FaultTo**.

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

509 **/wsa:Action**

510 **R0160** - this mandatory element of type **xs:anyURI** denotes the type of the OSCI message  
 511 and MAY carry one of the values outlined in the table below; further values MAY be defined  
 512 according to implementation needs. An OSCI message MUST carry exactly one  
 513 **/wsa:Action** SOAP header element.

<b>wsa:Action URIs assigned to OSCI Message Types</b>
<b>ht t p: // www. osci . eu/ ws/ 2008/ 05/ t r ansport / ur n/ messageTypes/ osci : Request</b>
<b>ht t p: // www. osci . eu/ ws/ 2008/ 05/ t r ansport / ur n/ messageTypes/ osci : Response</b>
<b>ht t p: // www. osci . eu/ ws/ 2008/ 05/ t r ansport / ur n/ messageTypes/ MsgBoxFet chRequest</b>
<b>ht t p: // www. osci . eu/ ws/ 2008/ 05/ t r ansport / ur n/ messageTypes/ MsgBoxSt at usLi st Request</b>
<b>ht t p: // www. osci . eu/ ws/ 2008/ 05/ t r ansport / ur n/ messageTypes/ MsgBoxResponse</b>
<b>ht t p: // www. osci . eu/ ws/ 2008/ 05/ t r ansport / ur n/ messageTypes/ MsgBoxGet Next Request</b>
<b>ht t p: // www. osci . eu/ ws/ 2008/ 05/ t r ansport / ur n/ messageTypes/ MsgBoxCl oseRequest</b>

514 Table 3: Predefined URIs for the WS Addressing Action element

515 If this header element carries a value not known to the node receiving the message, it MUST  
516 be discarded and a fault MUST be generated.

517 **Fault 2: AddrWrongActionURI**

518 [Code] Sender

519 [Subcode] AddrWrongActionURI

520 [Reason] Invalid Action header URI value

521 **/wsa:MessageID**

522 **R0170** - this mandatory element of type **wsa:AttributedURIType** MUST carry a unique  
523 message ID (UUID) according to IETF RFC "A Universally Unique Identifier (UUID) URN  
524 Namespace" [RFC4122]. An OSCI message MUST carry exactly one **/wsa:MessageID**  
525 SOAP header element.

526 **/wsa:RelatesTo \***

527 **R0180** - these optional elements of type **xs:anyURI** MUST be included, if a message is to  
528 be seen as a response to preceding messages and in this case MUST carry the  
529 **wsa:MessageID** SOAP header entry of those messages. This is always the case for the  
530 network backchannel **osci:Response** and **osci:MsgBoxResponse**. In case of asynchronous  
531 responses on Content Data level (carried in a new **osci:Request**) the values for these  
532 elements MUST be supplied by the responding Target Application. In case of an  
533 **OSCI fetched Notification** (see chapter [8.3.4]), the value MUST be the one of the message  
534 currently being fetched out of a **MsgBox** instance.

535 **/wsa:RelatesTo/@RelationshipType ?**

536 This optional attribute of type **xs:anyURI** SHOULD be omitted. Following the semantics of  
537 [WSA], the implied value of this attribute is  
538 "<http://www.w3.org/2005/08/addressing/reply>".

539 **/wsa:ReferenceParameters** (mapped to **osci:TypeOfBusinessScenario** in the SOAP  
540 binding)

541 **R0190** - an OSCI message MUST carry at least one element according to the SOAP  
542 mapping defined for **wsa:ReferenceParameters**. According to R0120, this is an URI  
543 carried in a SOAP header element **osci:TypeOfBusinessScenario** which is bound to  
544 the address element **wsa:To**. The SOAP header **osci:TypeOfBusinessScenario**  
545 MUST carry an attribute **wsa:IsReferenceParameter="true"**.

546 If this header element is missing or the addressed endpoint is not able to serve the concrete  
547 **osci:TypeOfBusinessScenario**, a fault MUST be generated and the message MUST  
548 be discarded:

549 **Fault 3: AddrWrongTypeOfBusinessScenario**

550 [Code] Sender

551 [Subcode] AddrWrongTypeOfBusinessScenario

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

## 553 6.2 Non addressable Initiators and use of WS MakeConnection

554 Non-addressable initiators themselves can create outbound connections but cannot accept  
555 connections from systems outside their network. This may be for reasons of network topology (i.e.  
556 NATs), security (i.e. firewalls), or the like. In the view of the OSCI topology, such initiators even have  
557 no **MsgBox** service available where asynchronous response messages can be targeted to. SOAP  
558 supports non-addressable clients by leveraging HTTP to take advantage of this fact. Non-addressable  
559 SOAP clients create an outbound connection to a server, send the request message over this

560 connection, then read the corresponding response from that same connection (this response channel  
561 is referred to as "the HTTP back-channel"). This is why non-addressable clients operate  
562 synchronously. The response can be delivered in the HTTP backchannel of the request. For this  
563 behaviour, WS-Addressing specifies the anonymous URI to be carried in the `/ReplyTo` EPR:  
564 `"http://www.w3.org/2005/08/addressing/anonymous"`.

565 For responses to be delivered to non-addressable initiators in an asynchronous way, the specification  
566 WS MakeConnection [WSMC] defines mechanisms to uniquely identify anonymous endpoints as well  
567 as making responses accessible for the initiator in a response pulling manner. On the recipient site  
568 special features have to be foreseen to hold responses until they are pulled. The fact a recipient  
569 endpoint serves (and in this also requires) support of the MakeConnection protocol and is indicated by  
570 a policy assertion as described in chapter [10.3].

571 OSCI implementations MAY support WS MakeConnection; no profiling applies here. Special attention  
572 should be taken here concerning the authentication requirements for anonymous initiators and  
573 message security to prevent unauthorized message access.

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

### 578 **6.3 Addressing faults**

579 The WS Addressing fault handling defined in [WSASOAP], chapter 6 "Faults" applies. For general fault  
580 handling, see chapter [5.2].

## 581 7 Message Security, Authentication and Authorization

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

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

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

- 593 • Possible use of transport layer mechanisms (HTTP over SSL/TLS); if useable the profiling of  
594 [WSI-BSP11], chapter 4 "Transport Layer Mechanisms" MUST be applied. For cryptographic  
595 suites using TLS the German BSI Technical Guideline [TR02102-2] applies.
- 596 • If message layer mechanisms must be used: Which message parts have to be encrypted and  
597 signed as well as security token to be used for these purposes?
- 598 • What kind of token for authentication and authorization must be provided in a message?

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

### 600 7.1 WS Security Header Block

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

- 604 • Transport encryption and signing is achieved by means defined in [XMLDSIG] and [XENC] for  
605 which a profiling is provided in the following subchapters [7.2] and [7.3]. As defined by security  
606 policies, signature and/or encryption application to message parts is outlined in chapter [10].
- 607 • Supported security token types, outlined in chapter [7.4].

608 WS Security defines a timestamp element for use in SOAP messages. OSCI places the following  
609 constraint on its use:

610 **R0200** - A SOAP header / **wsse: Security** MUST contain exactly one element  
611 /**wsu:Timestamp**. This supersedes R3227 of [WSI-BSP11].<sup>7</sup>

### 612 7.2 XML Digital Signature

#### 613 7.2.1 Restrictions to WS-I Basic Security Profiling

614 The profiling of [WSI-BSP11], chapter 9 "XML-Signature" MUST be applied with the following  
615 restrictions going beyond them<sup>8</sup>:

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

<sup>7</sup> "MUST NOT contain more than one" is profiled by [WSI-BSP11]

<sup>8</sup> Recommendation: These restrictions SHOULD be regarded by SAML-Token issuers, too.

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

Digest method algorithms
<code>ht t p: // www. w3. or g/ 2001/ 04/ xml enc#sha256</code>
<code>ht t p: // www. w3. or g/ 2001/ 04/ xml enc#sha512</code>

622 Table 4: Digest method: allowed algorithm identifiers

- 623 The use of SHA-256 (`ht t p: // www. w3. or g/ 2001/ 04/ xml enc#sha256`) as digest  
624 method algorithm is RECOMMENDED. This supersedes R5420 of [WSI-BSP11]<sup>9</sup>.

- 625 **R0320** - For asymmetric signature methods, the following algorithms MUST be used:

Asymmetric signature method algorithms
<code>ht t p: // www. w3. or g/ 2001/ 04/ xml dsi g- m0r e#r sa- sha256</code>
<code>ht t p: // www. w3. or g/ 2001/ 04/ xml dsi g- m0r e#r sa- sha512</code>
Symmetric signature method algorithms
<code>ht t p: // www. w3. or g/ 2001/ 04/ xml dsi g- m0r e#hmac- sha256</code>
<code>ht t p: // www. w3. or g/ 2001/ 04/ xml dsi g- m0r e#hmac- sha512</code>

626 Table 5: Signature method: allowed algorithm identifiers

- 627 This supersedes R5421 of [WSI-BSP11]<sup>10</sup>.

628 **Note:** To ensure downwards compatibility, the verification of signatures produced with  
629 `ht t p: // www. w3. or g/ 2001/ 04/ xml dsi g- m0r e#r sa- r i pemd160` MUST be supported.

630 **NOTE** on R0310, R0320: The URI-Values of the attributes `ds:SignatureMethod/@Algorithm`  
631 and `ds:DigestMethod/@Algorithm` are fixed to identifiers resulting from the actual list of strong  
632 hash algorithms published in [TR02102-2]. One of the values outlined above MUST be chosen. *This*  
633 *enumeration is subject to future changes, in case of one of the algorithms must be seen to get weak.*

## 634 7.2.2 Format of XML Digital Signatures used for Documents

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

638 Here, the national signature laws and ordinances must be considered; in consequence a profiling of  
639 relevant standards has already been derived as well as classification of applicability of cryptographic  
640 algorithms. This leads to a profiling of those XML Digital Signatures, which are applied on documents  
641 as advanced or qualified signatures using an appropriate X.509v3-Certificate.

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

- 643 **R0400** - The detached XML Signature format MUST be used and the signed content, if part of the  
644 message (child of SOAP envelope), be referenced by the local (fragment) URI mechanism  
645 as defined in [RFC2396]. Referenceable fragments of message parts MUST carry an  
646 attribute of type `xs:ID`. The constraints of the XML 1.0 [XML 1.0] ID type MUST be met.

<sup>9</sup> SHOULD is defined by [WSI-BSP11] for `ht t p: // www. w3. or g/ 2000/ 09/ xml dsi g#sha1`

<sup>10</sup> SHOULD is defined by [WSI-BSP11] for signature method algorithms based on SHA-1

- 647 The generation of unique ID attribute value SHOULD follow [RFC4122], this value  
648 SHOULD be concatenated to a preceding string "uuid:".<sup>11</sup>
- 649 **R0410** - A `ds:Signature` element MUST contain at least one `ds:Object` child element to carry  
650 the signing time and a reference to the certificate used for signing. The format of this child  
651 element MUST conform to definitions given by [XAdES] and the following restrictions must  
652 apply here:
- 653 It MUST contain exactly one child element `xades:QualifyingProperties` including  
654 the mandatory child element,  
655 `xades:SignedProperties/xades:SignedSignatureProperties` and an  
656 optional child element `xades:UnsignedProperties`, which is foreseen to carry a  
657 qualified timestamp over the signature itself in the child element  
658 `xades:UnsignedSignatureProperties/xades:SignatureTimeStamp`.
- 659 Child elements of `xades:SignedSignatureProperties` which MUST be present are  
660 `xades:SigningTime` and information about the certificate used for signing in  
661 `xades:SigningCertificate`.
- 662 **R0420** - As consequence of R0300 and R0310, a `ds:Signature` element MUST contain at least  
663 two `ds:Reference` child elements for referencing at least one detached content and the  
664 elements in `ds:Object` to be included in the signature calculation.
- 665 **R0430** - Exclusive canonicalization MUST be used to address requirements resulting from  
666 scenarios where subdocuments are moved between contexts. The URI-Value of the  
667 attribute `ds:CanonicalizationMethod/@Algorithm` is fixed to  
668 "`http://www.w3.org/2001/10/xml-exc-c14n#`".<sup>12</sup>
- 669 **R0440** - Signatures are only applicable based on X.509v3-Certificates which MUST conform to  
670 [COMPKI]. The child elements `ds:RetrievalMethod` and `ds:X509Data` of  
671 `ds:KeyInfo` MUST be present. All other choices according to [XMLDSIG] for  
672 `ds:KeyInfo` MUST NOT be present. In consequence, the attribute  
673 `ds:RetrievalMethod/@Type` MUST carry a value of  
674 "`http://www.w3.org/2000/09/xmlsig/X509Data`".
- 675 **R0450** - The child elements `ds:X509IssuerSerial` and `ds:X509Certificate` of  
676 `ds:X509Data` MUST be present; the child element `ds:X509CRL` SHOULD NOT be  
677 present to avoid significant data overload of signature elements to be expected in case of  
678 including CRLs. All other choices according to [XMLDSIG] for `ds:X509CRL` and  
679 `ds:X509SKI` MUST NOT be present.

680 **According to the XAdES Baseline Profile [XAdES-B], B-level Conformance**, profiling and restrictions  
681 are defined by the following normative outline:

```

682 <ds:Signature Id="xs:ID" >
683   <ds:SignedInfo Id="xs:ID" ?>
684     <ds:CanonicalizationMethod
685       Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"
686     </ds:CanonicalizationMethod>
687
688     <ds:SignatureMethod Algorithm=
689       "http://www.w3.org/2001/04/xmldsig-core#rsa-sha256" |
690       "http://www.w3.org/2001/04/xmldsig-core#rsa-sha512"
691     </ds:SignatureMethod>
692
693     <ds:Reference Id="xs:ID" ?
694       Type="http://uri.etsi.org/011903/v1.1.1/#SignedProperties"
695       URI="xs:anyURI" >
696     <ds:Transforms /> ?

```

<sup>11</sup> WS-Frameworks may foresee other ID attribute value generation mechanisms

<sup>12</sup> See also: R5404 of [WSI-BSP11]

```

697     <ds: DigestMethod Algorithm="
698       "http://www.w3.org/2001/04/xmlenc#sha256" |
699       "http://www.w3.org/2001/04/xmlenc#sha512"
700     </ds: DigestMethod>
701   <ds: DigestValue> xs:base64Binary </ds: DigestValue>
702   <ds: Reference>
703     ( <ds: Reference Id="xs:ID" ?
704       Type="xs:anyURI"
705       URI="xs:anyURI" >
706     <ds: Transform/> ?
707     <ds: DigestMethod Algorithm="
708       "http://www.w3.org/2001/04/xmlenc#sha256" |
709       "http://www.w3.org/2001/04/xmlenc#sha512"
710     </ds: DigestMethod>
711     <ds: DigestValue> xs:base64Binary </ds: DigestValue>
712     </ds: Reference> ) +
713
714
715 </ds: Signature>
716
717 <ds: SignatureValue Id="xs:ID" ?> xs:base64Binary </ds: SignatureValue>
718
719 <ds: KeyInfo Id="xs:ID" ?>
720   <ds: RetrievalMethod
721     Type="http://www.w3.org/2000/09/xmlenc#X509Data"/>
722   <ds: X509Data>
723     <ds: X509IssuerSerial>
724       <ds: X509IssuerName> xs:string </ds: X509IssuerName>
725       <ds: X509SerialNumber> xs:integer </ds: X509SerialNumber>
726     </ds: X509IssuerSerial>
727     <ds: X509Certificate> xs:base64Binary </ds: X509Certificate>
728     <ds: X509CRL/> ?
729   </ds: X509Data>
730 </ds: KeyInfo>
731
732 <ds: Object Id="xs:ID" >
733   <xades: QualifyingProperties Target="..." >
734     <xades: SignatureProperties>
735       <xades: SignatureProperties Id="xs:ID" >
736         <xades: SigningTime> xs:dateTime </xades: SigningTime>
737         <xades: SigningCertificate>
738           <xades: Cert>
739             <xades: CertDigest>
740               <ds: DigestMethod Algorithm="
741                 "http://www.w3.org/2001/04/xmlenc#sha256" |
742                 "http://www.w3.org/2001/04/xmlenc#sha512"
743               </ds: DigestMethod>
744               <ds: DigestValue> xs:base64Binary </ds: DigestValue>
745             </xades: CertDigest>
746             <xades: IssuerSerial>
747               <ds: X509IssuerName> xs:string </ds: X509IssuerName>
748               <ds: X509SerialNumber>
749                 xs:integer
750               </ds: X509SerialNumber>
751             </xades: IssuerSerial>
752           </xades: Cert>
753         </xades: SigningCertificate>
754       </xades: SignatureProperties>
755     </xades: SignatureProperties>
756
757     ( <xades: UnsignedProperties Id="xs:ID" ?>
758       <xades: UnsignedSignatureProperties Id="xs:ID" ?>
759         <xades: SignatureTime Id="xs:ID" ?>
760           ( <ds: CanonicalizationMethod
761             Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"
762           </ds: CanonicalizationMethod> ) ?
763         <xades: EncapsulatedSignature
764           Id="xs:ID" ? Encoding="xs:ID" ?>
765           xs:base64Binary
766         </xades: EncapsulatedSignature>
767         </xades: SignatureTime>

```

```

768     </ xades: Unsi gnedSi gnat ur ePr oper t i es>
769     </ xades: Unsi gnedPr oper t i es> ) ?
770
771     </ xades: Qual i f yi ngPr oper t i es>
772 </ ds: Obj ect >
773 <ds: Obj ect I d=" xs: I D" ?/ > ?
774 </ ds: Si gnat ur e>

```

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

777 For illustration, an example is given for an instance of such a signature element in Appendix C.

## 778 7.3 XML Encryption

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

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

### 784 7.3.1 End-to-end Encryption of Content Data

785 The following general rules apply in addition to those presented in chapter [7.3.2]:

786 **R0400** - If MsgBox service instances are involved on the message route, a SOAP message body  
 787 block MUST be encrypted for the intended Ultimate Recipient following [XENC] using the  
 788 public key of its X.509v3 encryption certificate. For other MEP's, encryption of the SOAP  
 789 body block is RECOMMENDED.

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

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

797

### 7.3.2 Encryption Cyphersuite Restrictions

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

Encryption Algorithm	Algorithm Identifier
AES-128-GCM	<a href="http://www.w3.org/2009/xmlenc11#aes128-gcm">http://www.w3.org/2009/xmlenc11#aes128-gcm</a>
AES-192-GCM	<a href="http://www.w3.org/2009/xmlenc11#aes192-gcm">http://www.w3.org/2009/xmlenc11#aes192-gcm</a>
AES-256-GCM	<a href="http://www.w3.org/2009/xmlenc11#aes256-gcm">http://www.w3.org/2009/xmlenc11#aes256-gcm</a>

800 Table 6: Symmetric encryption algorithms

801 **R0510** - Encryption of symmetric keys MUST be performed by means of RSAES-OAEP-ENCRYPT  
802 [PKCS#1]. The value of `xenc:EncryptionMethod` MUST be

803 `"http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p"`

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

## 7.4 Security Token Types

806 To be extensible, the WS-Security specification is not bound to specific security token types. For this  
807 version of OSCI Transport, token types outlined in following table MAY be used for authentication,  
808 message signature, and encryption operations. A profilings of those token types has been specified by  
809 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	<a href="http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0">http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0</a>	[WSSSAML]
X.509v3-Certificate	MUST	<a href="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3">http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3</a>	[WSSX509]
Kerberos	MAY	<a href="http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#GSS_Kerberosv5_AP_REQ">http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#GSS_Kerberosv5_AP_REQ</a>	[WSSKERB]
Username	MAY	Defined in WS-Security as <code>wsse:UsernameToken</code>	[WSSUSER]

810 Table 7: Security token types – support requirements

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

813 If access of anonymous initiators shall be supported, Public Key Infrastructure MUST be used  
814 applying X.509v3-Certificates:

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

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

821 Fault 4: **AuthnCertNotValid**

822 [Code] Sender

823 [Subcode] AuthnCertNotValid

824 [Reason] Authentication certificate not stated to be valid

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

828 **R0620** - X.509v3-Certificates used for authentication MUST have set the key usage extension to  
829 "digitalSignature". If the "nonRepudiation" key usage is set, these certificates MUST not  
830 be used for authentication.<sup>13</sup>

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

835 Fault 5: **AuthnCertInvalidKeyUsage**

836 [Code] Sender

837 [Subcode] AuthnCertInvalidKeyUsage

838 [Reason] Certificate not permitted for authentication

839 Token of type Username and Kerberos MAY be used for authentication and securing messages inside  
840 closed communication domains, where security and trust is given by means out of band of this  
841 specification.

## 842 7.5 Use of WS-Trust and SAML Token

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

845 **R0630** - Each access to an endpoint MUST be authorized by an STS instance of the endpoints  
846 Trust Domain. An STS MUST be able to confirm the requestors identity based on  
847 presented credentials.

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

852 **SAML Version 2.0 MUST be supported by OSCI implementations, SAML Version 1.1 is seen to be in**  
853 **a phasing out but SHOULD be supported if still used in certain Trust Domains.**

854 When dealing with the German eID infrastructure, the BSI "Technical Guideline TR-03130-1 eID-  
855 Server in part 1 [TR03130-1] defines the SAML-Token profile which have to be used; for an example  
856 application scenario specific profile see project [SAFE].

---

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

## 857 7.5.1 Authentication Strongness

858 Access authorization at least is given by the assurance of a certain level of authentication of the STR.  
859 Trustworthiness of the STR identity confirmation through an STS is given by the strongness of the  
860 following two processes:

- 861 • Initial registration of an endpoint at its IdP – organizational rules that applied for the degree of  
862 trustworthiness initial subject identification
- 863 • Mechanisms used for authentication at the time of requesting identity confirmation from the  
864 STS to match claimed and conformed identity.

865 [SAML1] respective [SAML2] and [SAMLAC] specify an authentication statement  
866 `saml<1|2>:AuthnStatement` to carry such information. Differentiated authentication context details  
867 may be included herein.

868 The BSI “Technical Guideline TR-03107-1 Electronic Identities and Trust Services in E-Government”  
869 in it’s part 1 [\[TR03107-1\]](#) defines assurance levels and mechanisms, which have to be seen  
870 compulsory for Germany<sup>14</sup>.

871 The [SAFE] project defined the following ascending levels for strongness of registration and  
872 authentication<sup>15</sup>:

- 873 • `urn:de:egov:names:fim:1.0:securitylevel:normal`
- 874 • `urn:de:egov:names:fim:1.0:securitylevel:high`
- 875 • `urn:de:egov:names:fim:1.0:securitylevel:veryhigh`

876 Each level matches operational rules which must be defined, published, and continuously maintained  
877 by appropriate institutions, i.e. government agencies concerned with data protection.<sup>16</sup>

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

```
880 <saml ac: Extension>
881   <fi mac: SecurityLevel >
882     <fi mac: Authentication>
883       urn:de:egov:names:fim:1.0:securitylevel:normal |
884       urn:de:egov:names:fim:1.0:securitylevel:high |
885       urn:de:egov:names:fim:1.0:securitylevel:veryhigh |
886     </fi mac: Authentication> ?
887   <fi mac: Registration>
888     urn:de:egov:names:fim:1.0:securitylevel:normal |
889     urn:de:egov:names:fim:1.0:securitylevel:high |
890     urn:de:egov:names:fim:1.0:securitylevel:veryhigh |
891   </fi mac: Registration> ?
892 </fi mac: SecurityLevel > ?
893 </saml ac: Extension> ?
```

894 **This outline is to be seen as informational, applies to scenarios building on [SAFE].**

895

<sup>14</sup> As of January 2015, assurance levels are about to be standardized at European level in context of the implementation of the “Regulation (EU) N°910/2014 on electronic identification and trust services for electronic transactions in the internal market” (eIDAS Regulation); respective outcomes will be adopted in a future revision of this specification.

<sup>15</sup> Preliminary URIs proposed by the SAFE-Project; subject to standardization activities by German administration.

<sup>16</sup> Definition of such rules cannot be a matter of this specification. An example for a level “veryhigh” could be a registration data confirmation based on presenting Id Cards and subsequent authentication using authentication certificates issued by accredited CAs.

896 `/samlac:Extension ?`

897 Optional container carrying the extension; to be included in a SAML assertion in the  
898 `saml ac: Auth hent i cat i onCont ext Decl ar at i on` element.

899 `.../fimac:SecurityLevel ?`

900 Optional container carrying the detail elements.

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

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

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

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

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

909 **Fault 6: AuthnSecurityLevelInsufficient**

910 [Code] Sender

911 [Subcode] AuthnSecurityLevelInsufficient

912 [Reason] Insufficient strongness of authentication or registration

913 Detailed information on the security level requirements SHOULD be provided in the fault [Details]  
914 property in this case.

915 To facilitate the acquisition of an appropriate SAML token for the initiator, endpoints SHOULD  
916 describe their requirements on authentication strongness by means of WS-Policy as will be outlined by  
917 concrete WSDL patterns published in 2009 as addendums to this document.

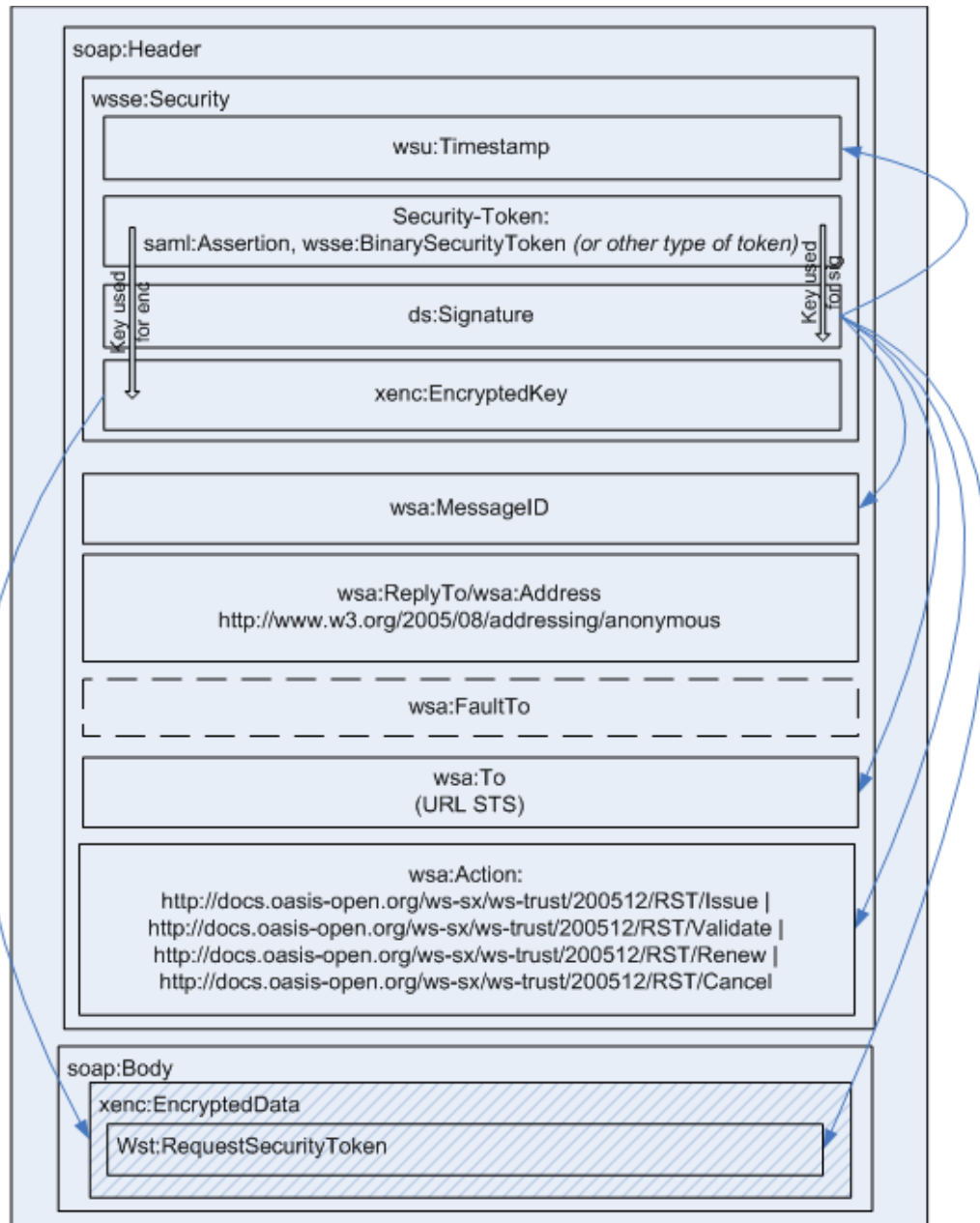
## 918 **7.5.2 WS-Trust Messages**

919 Conformant OSCI Gateway implementations MUST support the SOAP message types and bindings  
920 defined by WS-Trust:

- 921 • Issue
- 922 • Validate
- 923 • Cancel.

924 The WS-Trust Renew-Binding SHOULD be supported for convenience; this functionality is supplied by  
925 most STS-implementations.

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

930 **7.5.2.1 Request Security Token (RST)**

931

932

Figure 2: Request Security Token Message

933 SOAP header blocks:

934 **/wsse:Security**

935 This header block MUST be present, carrying message protection data and initiator  
 936 authentication information according the security policy of the STS the RST message is  
 937 targeted to.

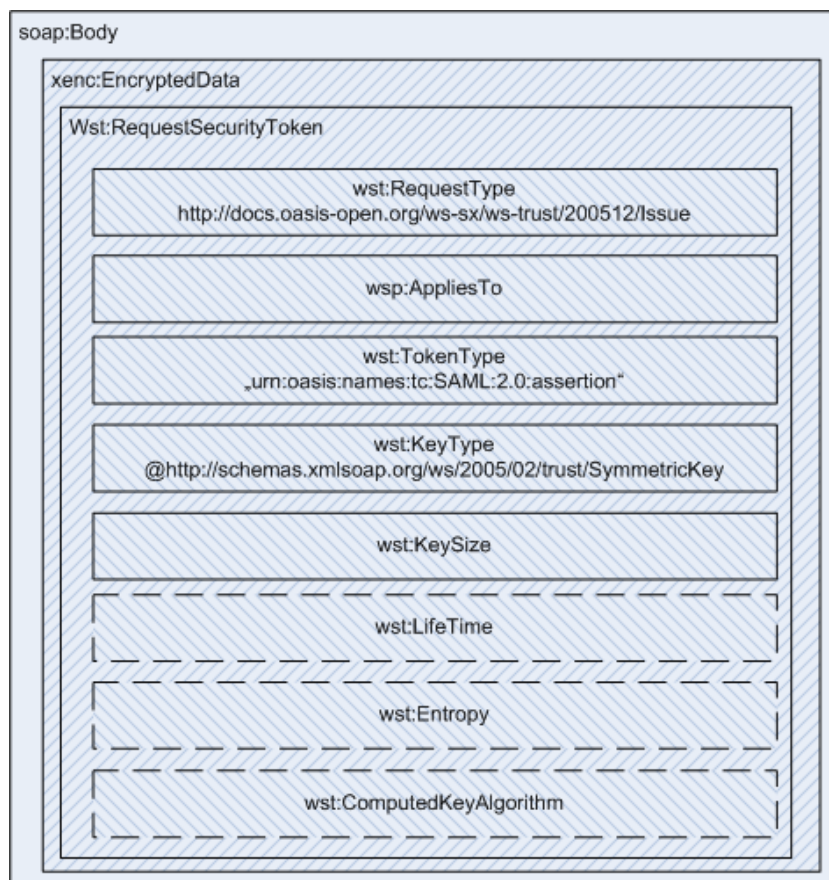
938 **/wsse:Security/wsue:Timestamp**

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

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

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

- 944 [Table 7] lists the security token types which MUST or MAY be supported.
- 945 Security tokens MUST be embedded or referenced. Referenced tokens MUST be  
946 dereferencable by the targeted STS.
- 947 The requestors security token MUST be used for signing the above marked message parts.
- 948 **/wsse:Security/ds:Signature**
- 949 A signature containing **ds:Reference** elements for all message parts marked above is to  
950 be included in the signature.
- 951 **/wsse:Security/xenc:EncryptedKey**
- 952 The RST contained in the SOAP body block MUST be encrypted for the targeted STS. This  
953 is a symmetric key, which MUST be encrypted with the public key of the STS X.509v3  
954 encryption certificate. Rules outlined in chapter [7.3] apply. It is assumed that the STS  
955 encryptions certificate is made available to all endpoints inside the STS Trust Domain out of  
956 band of this specification.
- 957 **/wsa:\***
- 958 All WS-Addressing headers MUST (if in continuous blocks) /MAY (if in dashed blocks) be  
959 supplied by the requestor.
- 960 **/wsa:Action**
- 961 Depending on the type of WS-Trust request, one of the URIs outlined above MUST be  
962 supplied. This URI MUST be adequate to the respective body block content.
- 963 The SOAP body block MUST conform to the definitions of WS-Trust, whereby the following restrictions  
964 and recommendations apply for the WS-Trust Issue request type.



965  
966  
967

Figure 3: Request Security Token, Body for Issue Request

968 **/wsp:AppliesTo**

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

971 **NOTE:** For ease of message exchange inside a Trust Domain, it is RECOMMENDED to  
972 choose an expression (i.e. URL pattern) accepted at least by a MsgBox instance for all  
973 recipients nodes using this MsgBox instance. This leverages the burden and overhead,  
974 which would be given by a **/wsp:AppliesTo** value assignment to a concrete recipient  
975 EPR.

976 **/wst:TokenType**

977 **R0700:** This element MUST be present; the value MUST be a SAML V1.1 or V2.0 assertion  
978 type:

979 `urn:oasis:names:tc:SAML:2.0:assertion` |  
980 `urn:oasis:names:tc:SAML:1.0:assertion`

981 **/wst:KeyType**

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

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

984 **/wst:KeySize**

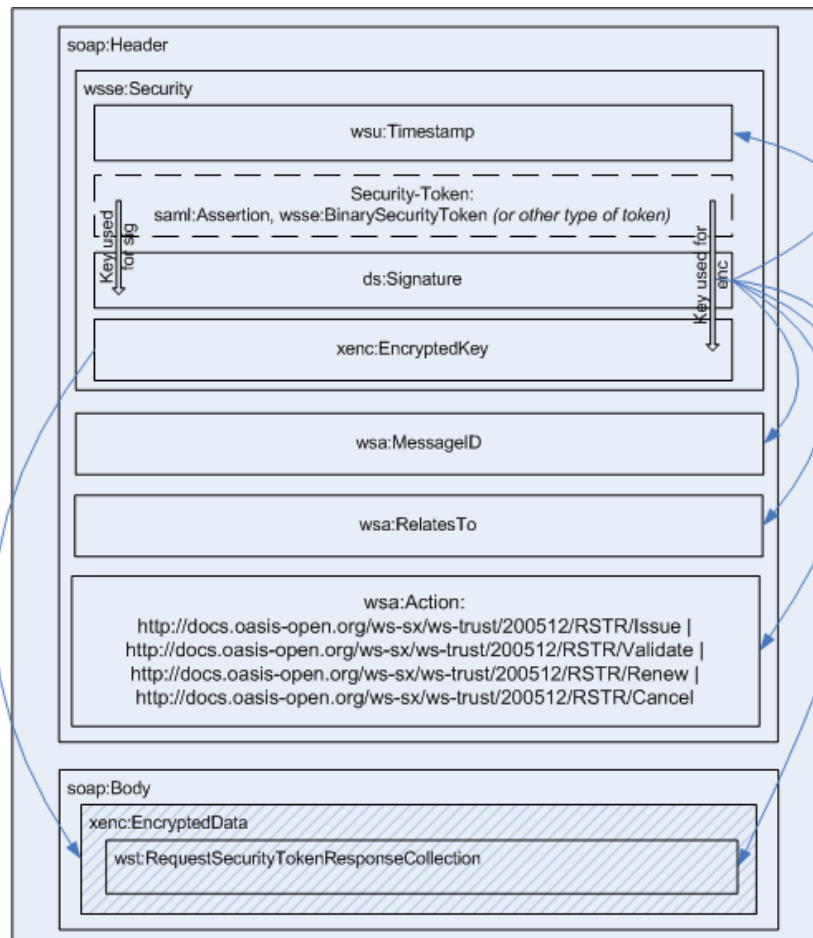
985 **R0720:** This element MUST be present; the key size MUST be greater or equal 256 Bit.

986 Use and values of elements marked optional are subject to used STS instance specific policies.  
987 Recommendations will we given as part of the amendments to be worked out for this specification in  
988 the future according requirements of concrete application scenarios.

989

990 **7.5.2.2 Request Security Token Response (RSTR)**

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



992

993 Figure 4: Request Security Token Response Message

994 Differences to the RST message:

995 **/wsse:Security/xenc:EncryptedKey**

996 The RSTRC contained in the SOAP body block MUST be encrypted for the token requestor.

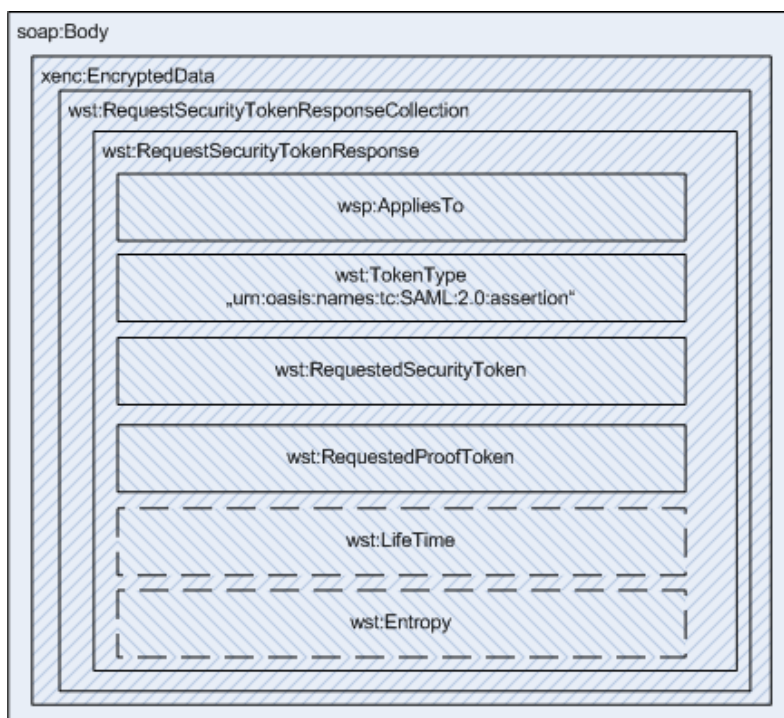
997 This is a symmetric key which MUST be encrypted with the public key of the requestors

998 X.509v3 encryption certificate. Rules outlined in chapter [7.3] apply.

999 **/wsa:\***1000 All WS-Addressing headers MUST (if in continuous blocks) /MAY (if in dashed blocks) be  
1001 supplied by the STS.1002 **/wsa:Action**1003 Depending on the type of WS-Trust response, one of the URIs outlined above MUST be  
1004 supplied. This URI MUST be adequate to the respective body block content.1005 The decrypted SOAP body block MUST conform to the definitions of WS-Trust. No restrictions or  
1006 profiling apply.

1007

1008 The SOAP body block MUST conform to the definitions of WS-Trust:



1009

1010

Figure 5: Request Security Token, Body for Issue Response

1011 Short description of the constituents of `/wst:RequestSecurityTokenResponse`, which is always  
 1012 wrapped by a `/wst:RequestSecurityTokenResponseCollection` (see [WST] for details):

1013 **`/wsp:AppliesTo`**

1014 Carries the value assignment for the desired application scope of the requested security  
 1015 token – copied from the according request element.

1016 **`/wst:TokenType`**

1017 Carries the token type, which MUST be the one of the according request elements.

1018 **`/wst:RequestedSecurityToken`**

1019 Carries the requested SAML-Token, including a symmetric key encrypted for the endpoint at  
 1020 which the SAML-Token is needed for authentication purposes. Details are explained in  
 1021 chapter [7.5.3].

1022 **`/wst:RequestedProofToken`**

1023 Carries information enabling the requestor to deduce the symmetric key. In case the key was  
 1024 generated by the STS solely, this is the key itself.

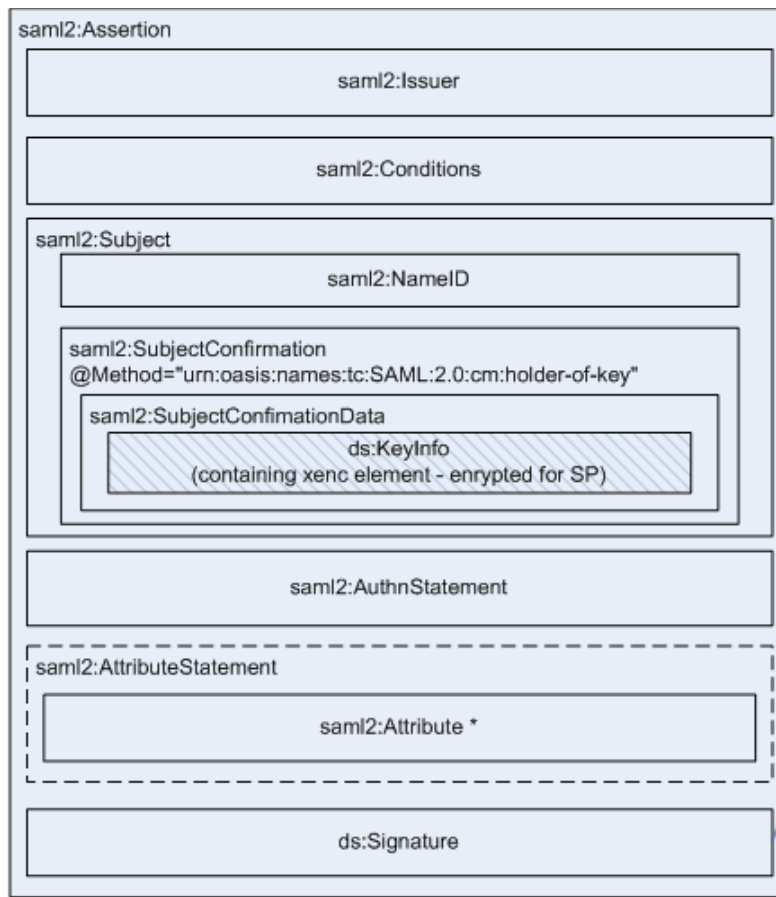
1025 In case computed of two entropy values, this is the algorithm and the element

1026 **`/wst:Entropy ?`**

1027 MUST be present carrying the entropy value used by the STS for key computation.

1028 **`/wst:LifeTime ?`**

1029 Optional element carrying the validity duration period of this RSTR; SHOULD be recognized  
 1030 by the requestor and processed according to his needs to avoid using security tokens  
 1031 being/getting invalid.

1032 **7.5.3 Issued SAML-Token Details**1033  
1034 Figure 6: SAML 2.0 Assertion constituents

1035 Short description of the constituents of a `/saml2:Assertion`, for XML Infoset details see [SAML2]  
 1036 <sup>17</sup>and [SAMLAC]. The concrete token request requirements and layout of issued token at least has to  
 1037 be matched with the capabilities of the used STS instances. For implementations to be operated in  
 1038 context of the German administration it is strongly RECOMMENDED to follow requirements and  
 1039 recommendations given by the concept [SAFE].

1040 `/saml2:Issuer`

1041 Attributes of the STS issuing this assertion; for details see `saml2:NameIDType`

1042 `/saml2:Conditions`

1043 **R0730:** Detailed validity conditions element MUST be present; for details see  
 1044 `saml2:ConditionsType`. MUST at least outline the validity period attributes  
 1045 `NotBefore`, `NotOnOrAfter`.

1046 `/saml2:Subject`

1047 **R0740:** Presence of this element is REQUIRED. The subelements of this container provide  
 1048 STR identification details.

1049 `/saml2:Subject/saml2:NameID`

1050 Attributes of the STR; for details see `saml2:NameIDType`. It MUST at least contain a  
 1051 unique string identifying the STR.

1052 `/saml2:Subject/saml2:SubjectConfirmation`

<sup>17</sup> For brevity, we only illustrate the SAML Version 2.0 Assertion in this document. For the SAML Version 1.1 Assertion layout, see [SAML1].

- 1053 This container exposes STS information for the SP enabling it to assure that the SR is the  
1054 one stated in `/saml2:NameID` and authorized to use this token.
- 1055 `/saml2:Subject/saml2:SubjectConfirmation/@Method`
- 1056 **R0750:** Attribute outlining the confirmation method; MUST be the "holder of key"  
1057 confirmation method.
- 1058 `/saml2:Subject/saml2:SubjectConfirmation/saml12:SubjectConfirmationData`
- 1059 **R0760:** Presence of this element is REQUIRED; it exposes STS information for the SP  
1060 enabling it to assure that the SR is the one owning the key for this SAML assertion.
- 1061 `/saml2:Subject/saml2:SubjectConfirmation/saml12:SubjectConfirmationData/ds:`  
1062 `key`
- 1063 **R0770:** This element MUST carry the key in a `xenc:EncryptedKey` element. The key  
1064 MUST be encrypted for the SP using the public key of its X.509v3 encryption certificate,  
1065 which for this purpose MUST be made available to the STS.
- 1066 NOTE on the endpoint encryption certificate, the SAML token is targeted to:
- 1067 The access to this certificate through the token issuing STS is of band of this specification;  
1068 this is a matter of Trust Domain policies and an implementation issue which MUST have no  
1069 effect on interoperability. No standardized mechanisms are foreseen by WS-Trust, to include  
1070 a certificate in an RST message for the purpose of key encryption for the SP. It is strongly  
1071 RECOMMENDED, to relate the `/wsp:AppliesTo` request value (which might be a pattern,  
1072 too – see RST body description in chapter [7.5.2.1]) to this encryption certificate.
- 1073 `/saml2:AuthnStatement`
- 1074 **R0780:** Presence of this element is REQUIRED.
- 1075 It MUST contain an element `/saml2:AuthnContext` with an attribute `@AuthnInstant`  
1076 outlining the time instant the authentication took place.
- 1077 `/saml2:AuthnContext` MUST contain an element `/saml2:AuthnContextClassRef`  
1078 outlining the authentication method used by the SR.<sup>18</sup>
- 1079 `/saml2:AuthnContext` MUST further contain an element `/saml2:AuthnContextDecl`  
1080 carrying the extensions for authentication strongness as defined in chapter [7.5.1].
- 1081 `/saml2:AttributeStatement ?`
- 1082 Usage of attribute statements of type `saml12:AttributeType` is RECOMMENDED. In  
1083 many scenarios subject attributes like affiliation to certain groups or roles are used for the  
1084 assignment's detailed rights, functions and data access. Hence attributes are specific to  
1085 application scenarios, their names, values, and semantics are subject to the overall design of  
1086 a domain information model, which is not addressed by this specification.<sup>19</sup>
- 1087 `/ds:Signature`
- 1088 The issuing STS has to sign the whole SAML-Token.
- 1089

<sup>18</sup> 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`

<sup>19</sup> Suggestions for use in German e-government, especially e-justice, are made in [SAFE].

1090 If a SAML token does not match one or more of the formal requirements 0730-0780, the token  
1091 consuming node MUST generate a fault and discard the message.

1092 **Fault 7: AuthnTokenFormalMismatch**

1093 [Code] Sender

1094 [Subcode] AuthnTokenFormalMismatch

1095 [Reason] Authentication token present does not match formal requirements.

1096 More information MAY be given in the fault [Details] property, but care should be taken to introduce  
1097 security vulnerabilities by providing too detailed information.

#### 1098 **7.5.4 Authentication for Foreign Domain Access**

1099 To authenticate and authorized access to foreign TD endpoints, these endpoints MUST be able to  
1100 validate the SAML-Token contained in the message. The specification WS-Federation 1.2 ([WSF],  
1101 chapter 2.4) outlines several possible trust topologies; for simplification, two of those described below  
1102 are selected to be applicable for this version of the OSCI specification. So far, the WS Federation  
1103 metadata model is not yet been taken into account for usage in OSCI 2 based infrastructures.

1104 Precondition for cross domain message exchange is an established trust relationship between the  
1105 initiator's STS and the one of the foreign TD. This i.e. can be achieved by trust in the STS signature  
1106 using its signing certificate as a trust anchor.

1107 One useable trust model is, a SAML-Token issued by the foreign STS must be acquired for accessing  
1108 endpoints in this TD. Authentication at a foreign STS in this case is obtained based on presenting the  
1109 SAML-Token of the initiators STS in the according RST issue message. Depending on policies in  
1110 effect, this SAML-Token may be replaced or cross-certified by applying a new signature. The SAML-  
1111 Token key MUST be encrypted for the endpoint access it is intended for. At the endpoint accessed,  
1112 SAML-Token validation can be done based on the signature of the foreign TD STS.

1113 In the second trust model, the SAML-Token issued by the initiator's STS directly and is used for  
1114 access authentication. In this case, the foreign endpoint points an RST validate message to his trusted  
1115 STS for validating the foreign SAML-Token – what is done there again based on SAML-Token  
1116 signature, which must be trusted by the validating STS. Again, the SAML-Token key MUST be  
1117 encrypted for the endpoint access it is intended for.

1118 Details of the required SAML Token including claims and the issuing STS address as well as the  
1119 public key of this STS encryption certificate SHOULD be exposed by the endpoint WSDL. Apart,  
1120 means of WS-Trust as already outlined in the chapters above apply.

#### 1121 **7.5.5 SAML-Token for Receipt/Notification Delivery**

1122 Requested receipts and notifications which cannot be delivered in the network backchannel of a  
1123 request message MUST be delivered using an independent request message asynchronously to the  
1124 EPR, outlined in the receipt/notification request – which in general SHOULD be the initiators MsgBox.  
1125 As – like for all messages - delivery of receipts/notifications to this EPR requires authentication and  
1126 authorization, an according SAML-Token SHOULD be forwarded to the receipt/notification generating  
1127 node together with the request for it. This mechanism disburdens these nodes from the acquisition of  
1128 an extra SAML-Token to authenticate receipt/notification delivery.

1129 This type of SAML-Token - referred to as "**OneTimeToken**" - is valid only for "one time use" of  
1130 receipt/notification delivery and bound to the `wsa:MessageID` of the message to be  
1131 receipted/notified. Following rules apply:

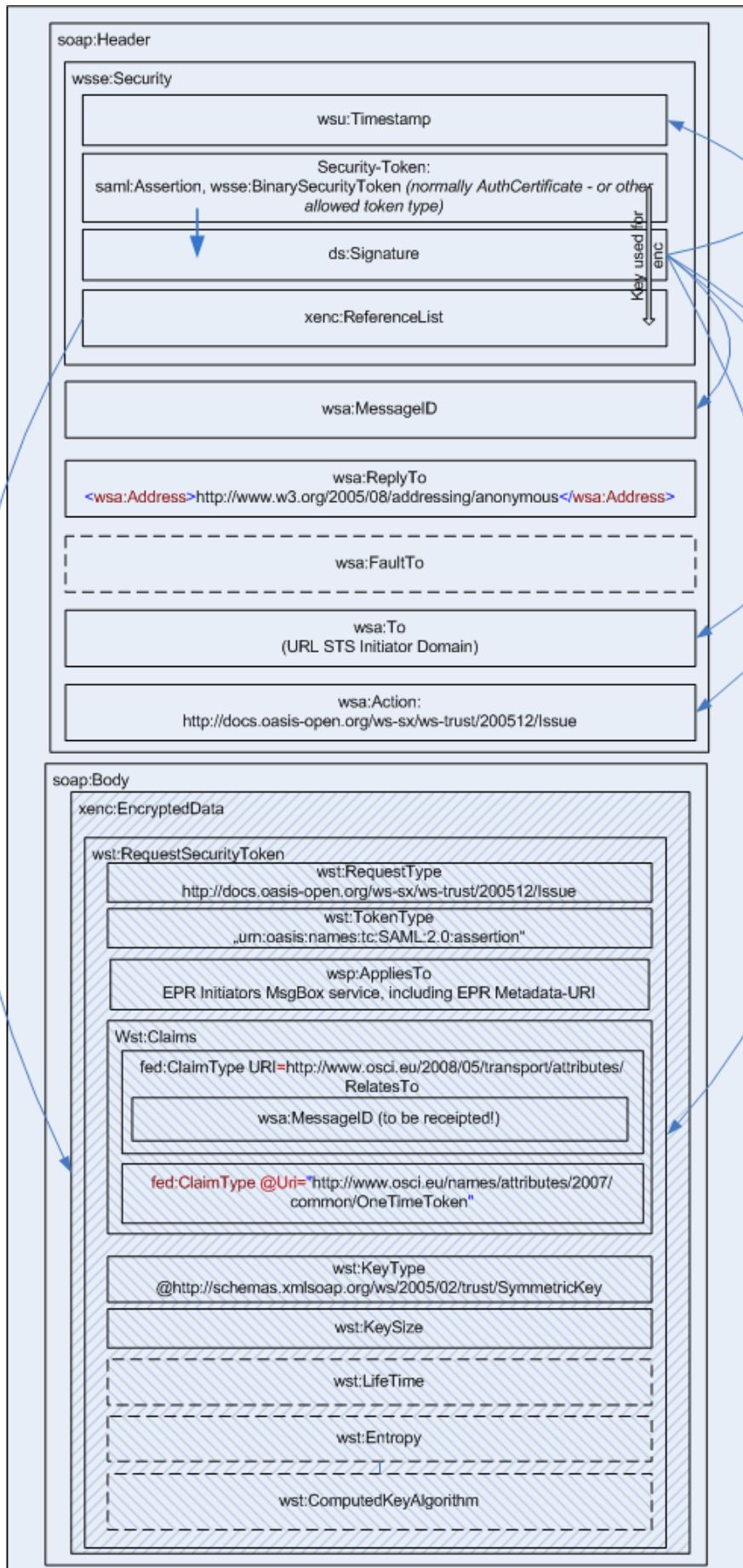
- 1132 1. It MUST be requested from the initiator's STS.
- 1133 2. The according RST message MUST contain the `wsa:MessageID` and the address of the  
1134 receipting/notifying node (`wsp:AppliesTo`) as claims.

- 1135 3. The symmetric key of the issued SAML-Token MUST be encrypted for the endpoint outlined in  
1136 the element `.../wsa:ReplyTo` of the receipt/notification demand; the  
1137 `wst:RequestedProofToken` in this case MUST be encrypted for the receipting/notifying  
1138 node (for use in the following step 6)
- 1139 4. The issuing STS MUST retain this OneTimeToken for later use and mark it as "unused".
- 1140 5. The RSTR message returned by the STS MUST be included as separate SOAP header block  
1141 in the request message.
- 1142 6. The receipting/notifying node has to use the OneTimeToken included in this RSTR as SAML-  
1143 Token for the message the receipt/notification is delivered with. Transport signature and  
1144 encryption MUST be generated based on the symmetric key contained in the  
1145 `wst:RequestedProofToken`.

1146 Steps to be done by the node, this message is targeted to:

- 1147 7. Decryption of the OneTimeToken's symmetric key.
- 1148 8. Validation of the signature of the OneTimeToken – the symmetric key MUST be the same the  
1149 receipting/notifying node used for the transport signature.
- 1150 9. Validation of the signature of the issuing STS and RST-Validate message containing the  
1151 OneTimeToken to the STS.
- 1152 10. If at the issuing STS this OneTimeToken is still marked as "unused", the token is valid.
- 1153 11. If the RSTR signals valid in validate-response: Acceptance of the message containing the  
1154 receipt/notification.
- 1155 12. Message accepting node MUST target a RST/cancel message to the STS to invalidate this  
1156 OneTimeToken; STS SHOULD discard this token.

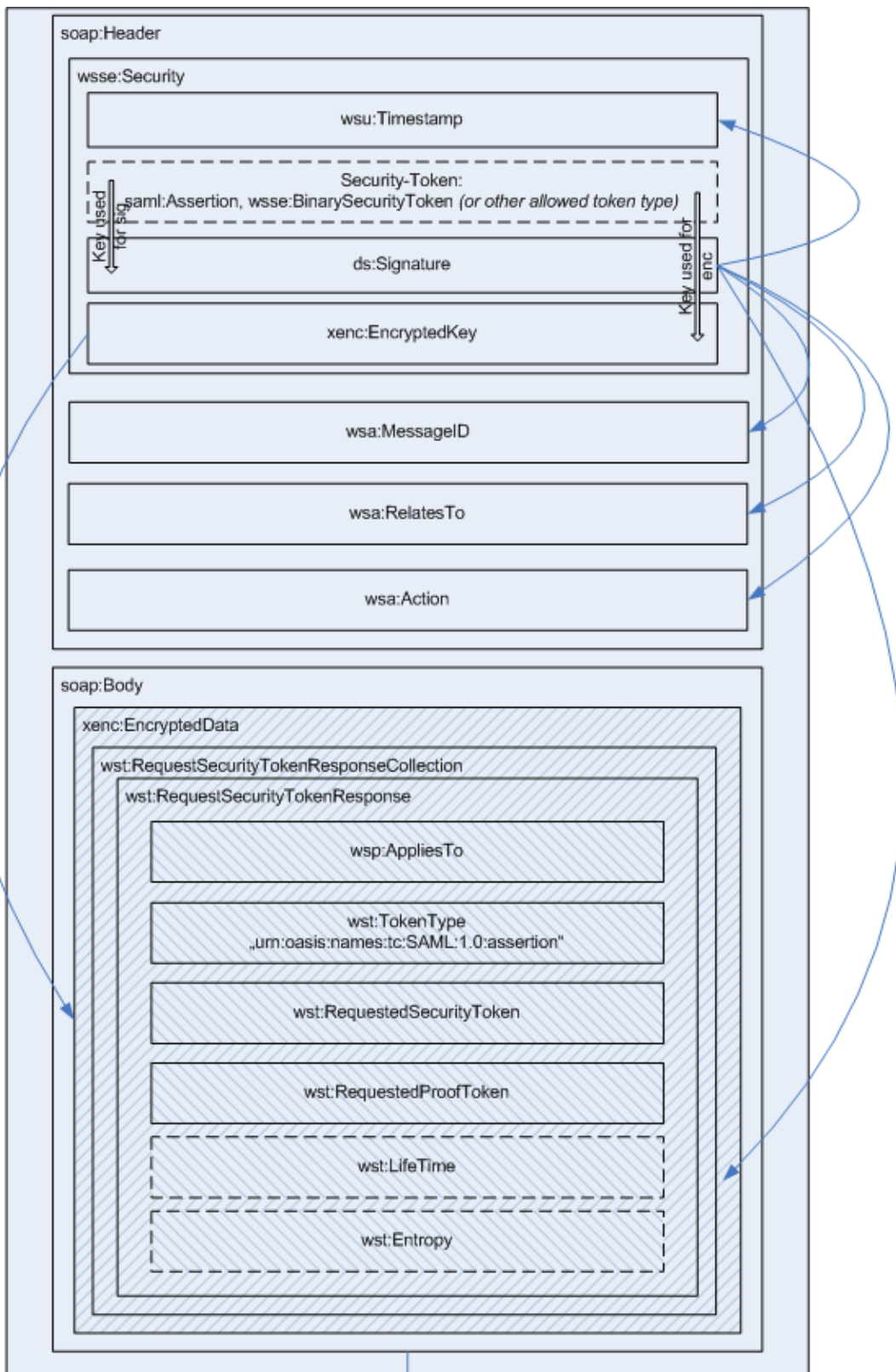
1157 The following diagrams illustrate the RST and RSTR for the OneTimeToken, for concrete XML Infoset  
1158 descriptions see WS-Trust and SAML specifications.



1159

1160

Figure 7: RST for OneTimeToken



1161  
1162

Figure 8: RSTR for OneTimeToken

## 1163 8 OSCI Specific Extensions

### 1164 8.1 Message Flow Time Stamping

1165 For sake of traceability of message flow time instants and delivery status,  
 1166 every message of type `osci:Request` MAY contain following SOAP header block,  
 1167 which child elements are provided depending on the nodes passed in the  
 1168 message flow. `<osci:MsgTimeStamps wsu:Id="..." ? >`  
 1169 `<osci:ObsoleteAfter> xs:date </osci:ObsoleteAfter ?`  
 1170 `<osci:Delivery> xs:dateTime </osci:Delivery ?`  
 1171 `<osci:InitialFetch> xs:dateTime </osci:InitialFetch ?`  
 1172 `<osci:Reception> xs:dateTime </osci:Reception ?`  
 1173 `</osci:MsgTimeStamps>`

1174 Description of elements and attributes in the schema overview above:

1175 **NOTE:** Elements of `osci:MsgTimeStamps` MUST NOT be provided or changed by other nodes on  
 1176 the message path than described here.

1177 `/osci:MsgTimeStamps`

1178 This complex element is the container for various optional timestamp elements. It MUST be  
 1179 created from the first node on the message flow which applies one or more sub-elements.

1180 `/osci:MsgTimeStamps/@wsu:Id`

1181 For ease of referencing this SOAP header block from WS Security SOAP header elements,  
 1182 this attribute of type `wsu:Id` SHOULD be provided.

1183 `/osci:MsgTimeStamps/osci:ObsoleteAfter ?`

1184 This element of type `xs:date` MAY be provided by an initiator to denote the date after  
 1185 which a message is to be seen as obsolete for delivery and/or consumption.

1186 If and how this information is handled by this endpoint this message is targeted to if outlined  
 1187 in the policy of this endpoint; see chapter [10.2.2] for details.

1188 `/osci:MsgTimeStamps/osci:Delivery ?`

1189 This element of type `xs:dateTime` MUST be provided by a recipient (synchronous MEP) or  
 1190 MsgBox node when accepting an incoming message and MUST be set to the value of the  
 1191 actual time.

1192 `/osci:MsgTimeStamps/osci:InitialFetch ?`

1193 This element of type `xs:dateTime` MUST be provided by a MsgBox node with the value of  
 1194 the actual MsgBox server time when an authorized recipient initially pulls the message from  
 1195 his MsgBox instance and commits the successful initial reception of this message. This  
 1196 SHOULD be done by a recipient after the first successful pulling of the message from his  
 1197 MsgBox.

1198 This element MUST NOT be updated during subsequent pull processing on the same  
 1199 message.

1200 `/osci:MsgTimeStamps/osci:Reception ?`

1201 This element of type `xs:dateTime` MAY be set by (or triggered through) a reader to his  
 1202 actual system time when successfully accepting an incoming message, but it should be  
 1203 considered that the signature is invalidated which was applied over SOAP header and body  
 1204 elements by the message issuing instance.

1205 It MUST be set by a MsgBox node to its actual server time when the Recipient commits the  
 1206 reception of a message through a `MsgBoxGetNextRequest` or `MsgBoxCloseRequest`.

## 1207 8.2 Accessing Message Boxes

1208 The following chapters define how to access MsgBox services for searching and pulling out messages  
 1209 as well as how to gain status lists describing content of message boxes. Statuses of those requests  
 1210 are delivered in the SOAP header block of the correlating responses, while the pulled messages  
 1211 respective status lists are delivered in the SOAP body block.

1212 At first we describe the requests, followed by the respective responses and additional messages to  
 1213 model "get next", "commit", and "close" semantics for iterative MsgBox access sequences.

1214 **NOTE:** To leverage implementation efforts, in aberration to foregoing versions of OSCI 2 transport  
 1215 specifications, MsgBox service implementations are not obligated to support all search criteria for  
 1216 messages as described below. If search expressions present in `osci:MsgSelector` are not  
 1217 supported, the according request MUST be discarded and a fault MUST be generated:

### 1218 Fault 8: `MsgSelectorNotSupported`

1219 [Code] Sender

1220 [Subcode] `MsgSelectorNotSupported`

1221 [Reason] Presented selection criteria not supported

1222 The selection expression leading to this fault SHOULD be provided in the fault [Details] property in this  
 1223 case.

### 1224 8.2.1 `MsgBoxFetchRequest`

1225 To request a message from an endpoint, a recipient MUST send a `MsgBoxFetchRequest` message to  
 1226 his MsgBox instance endpoint.

1227 The normative outline for a `MsgBoxFetchRequest` request is:

```

1228 <s12:Envelope ... >
1229   <s12:Header ... >
1230     ...
1231     <wsa:Action>
1232     http://www.osci.eu/ws/2008/05/transport/urn/messageTypes/MsgBoxFetchRequest
1233     </wsa:Action>
1234     <wsa:MessageID>xs:anyURI </wsa:MessageID>
1235     <wsa:To>xs:anyURI </wsa:To>
1236     <osci:TypeOfBusinessScenario wsa:IsReferenceParameter="1">
1237     xs:anyURI
1238     </osci:TypeOfBusinessScenario>
1239     ...
1240   </s12:Header>
1241   <s12:Body ... >
1242     <osci:MsgBoxFetchRequest MsgPart=("Envelope"|"Body"|"Header")? >
1243       <osci:MsgSelector newEntry=("true"|"false")>
1244         <wsa:MessageID> xs:anyURI </wsa:MessageID> *
1245         <wsa:RelatesTo> xs:anyURI </wsa:RelatesTo> *
1246         <osci:MsgBoxEntryTimeFrom>
1247           xs:dateTime
1248         </osci:MsgBoxEntryTimeFrom? >
1249         <osci:MsgBoxEntryTimeTo> xs:dateTime </osci:MsgBoxEntryTimeTo? >
1250         <osci:Extension> xs:anyType </osci:Extension? >
1251       </osci:MsgSelector? >
1252     </osci:MsgBoxFetchRequest >
1253   </s12:Body>
1254 </s12:Envelope>
  
```

1255 The following describes normative constraints on the outline listed above:

1256 `/s12:Envelope/s12:Header/wsa:Action`

1257 The value indicated herein MUST be used for that URI.

- 1258 `/s12:Envelope/s12:Header/wsa:MessageID`
- 1259 The request MUST carry a unique WS-Addressing MessageID.
- 1260 `/s12:Envelope/s12:Header/wsa:To`
- 1261 The address of the MsgBox (request destination) endpoint.
- 1262 `/s12:Envelope/s12:Header/osci:TypeOfBusinessScenario`
- 1263 This value of the instantiation of `/wsa:ReferenceParameters` MUST be supplied for this  
 1264 message type. The value of `/osci:TypeOfBusinessScenario` is taken as message  
 1265 selection argument and SHOULD match one of those accepted by this endpoint. If a  
 1266 MsgBoxFetchRequest contains no other arguments for message selection in the SOAP body  
 1267 element `osci:MsgBoxFetchRequest/osci:MsgSelector`, the messages to be  
 1268 selected MUST be those which have not yet been fetched. Those are all messages in the  
 1269 addressed MsgBox which have no SOAP header element or a value of zero in the time  
 1270 instant element `.../osci:MsgTimeStamps/osci:InitialFetched`. They MUST be  
 1271 delivered one per request-/ response in a FIFO-manner to the endpoint denoted by  
 1272 `/s12:Envelope/s12:Header/wsa:ReplyTo`.
- 1273 `/s12:Envelope/s12:Header/osci:TypeOfBusinessScenario/`  
 1274 `@wsa:IsReferenceParameter`
- 1275 In the following WS-Addressing, the element MUST be attributed with  
 1276 `@wsa:IsReferenceParameter="1"`
- 1277 The body of this message contains the actual request in a structure
- 1278 `/s12:Envelope/s12:Body/osci:MsgBoxFetchRequest`
- 1279 Container holding detailed selection arguments in addition to  
 1280 `/s12:Envelope/s12:Header/osci:TypeOfBusinessScenario` above; this element  
 1281 MAY be empty if no further selection criteria shall be provided.
- 1282 `/s12:Envelope/s12:Body/osci:MsgBoxFetchRequest/@MsgPart`
- 1283 This optional attribute of type `xs:NMTOKEN` allows the recipient to denote which part of a  
 1284 message shall be returned in the SOAP body of the subsequent MsgBoxResponse  
 1285 messages:
- 1286 • "Envelope" – returns the whole `s12:Envelope` container of the selected messages as  
 1287 child element of the SOAP body block of the response message.
  - 1288 • "Header": whole resulting SOAP header elements are included as child elements of  
 1289 the SOAP body block of the response message.
  - 1290 • "Body": only the original SOAP body child element<sup>20</sup> MUST be included unchanged as  
 1291 child element of the SOAP body of the response message.
- 1292 **NOTE:** This attribute has been introduced with version 2.0.1 of this specification.
- 1293 `/s12:Envelope/s12:Body/osci:MsgBoxFetchRequest/osci:MsgSelector ?`
- 1294 If this optional element is present, arguments of the attribute `@newEntry` and sub-elements  
 1295 MsgSelector MUST be processed as logical AND (after first OR-processing of the  
 1296 sequences of MessageIDs in the SOAP body elements `.../osci:MessageID` and  
 1297 `.../osci:RelatesTo`, if present).
- 1298 `/s12:Envelope/s12:Body/osci:MsgBoxFetchRequest/osci:MsgSelector/@newEntry ?`

<sup>20</sup> As recommended in chapter [5], a SOAP body is assumed to carry only one child element.

1299 This optional Boolean attribute is defaulted to the value "true", if not present. If present, this  
 1300 attribute denotes whether only already pulled or new entered messages have to be selected  
 1301 from the MsgBox. "New" messages are indicated by having no SOAP header element or the  
 1302 value "zero" in the time instant element  
 1303 `.../osci:MsgTimeStamps/osci:InitialFetched.`

1304 `/s12:Envelope/s12:Body/osci:MsgBoxFetchRequest/osci:MsgSelector/  
 1305 osci:MessageID *`

1306 If present, this element contains a sequence of WS-Addressing MessageIDs. By including  
 1307 this element, the request of a MsgBox service MUST limit its search to just those messages  
 1308 with these values in the WS-Addressing SOAP header element `.../wsa:MessageID.`

1309 `/s12:Envelope/s12:Body/osci:MsgBoxFetchRequest/osci:MsgSelector/  
 1310 osci:RelatesTo *`

1311 If present, this element contains a sequence of WS-Addressing MessageIDs. By including  
 1312 this element, the request of a MsgBox service MUST limit its search to just those messages  
 1313 with these values in the WS-Addressing SOAP header elements `.../wsa:RelatesTo.`

1314 `/s12:Envelope/s12:Body/osci:MsgBoxFetchRequest/osci:MsgSelector/  
 1315 osci:MsgBoxEntryTimeFrom ?`

1316 If present, this element denotes a value of type `xs:dateTime` as lower value when a  
 1317 message has been accepted by a MsgBox service. The resulting search expression is  
 1318 `.../osci:MsgBoxEntryTimeFrom >=` the value of `.../osci:Delivery` in the message  
 1319 SOAP header block `osci:MsgTimeStamps` if the correlated element  
 1320 `.../osci:MsgBoxEntryTimeTo` is not present in `.../osci:MsgSelector.`

1321 `/s12:Envelope/s12:Body/osci:MsgBoxFetchRequest/osci:MsgSelector/  
 1322 osci:MsgBoxEntryTo ?`

1323 If present, this element denotes a value of `xs:dateTime` as upper value when a message  
 1324 has been accepted by a MsgBox service. The resulting search expression is  
 1325 `.../osci:MsgBoxEntryTimeTo <=` the value of `.../osci:Delivery` in the message SOAP  
 1326 header block `osci:MsgTimeStamps` if the correlated element  
 1327 `.../MsgBoxEntryTimeFrom` is not present in `.../osci:MsgSelector.`

1328 If latter elements are both set, the resulting search expression is  
 1329 `.../osci:MsgBoxEntryFrom >= .../osci:Delivery <= .../osci:MsgBoxEntryTimeTo;`  
 1330 the value of `.../osci:MsgBoxEntryFrom` MUST be less or equal to the value of  
 1331 `.../osci:MsgBoxEntryTo.`

1332 `/s12:Envelope/s12:Body/osci:MsgBoxFetchRequest/osci:MsgSelector/  
 1333 osci:Extension/{any} *`

1334 This is an extensibility mechanism to allow other search criteria to be passed. For example,  
 1335 an XPath query could be used to search for messages that match a certain pattern.  
 1336 Implementations may use this element for defining search criteria on agreements outbound  
 1337 to this specification.

1338 **NOTE:** For implementations implementing version 2.0.1 and higher of this specification, it is  
 1339 **RECOMMENDED** to support XPath-querying based on the header Block  
 1340 `oscimeta:MessageMetaData` introduced with version 2.0.1 of this specification.

1341 Upon receipt and authentication of this message, the MsgBox service MUST locate any message that  
 1342 matches the selection criteria. Only messages originally targeted to this EPR MUST be returned. The  
 1343 search criteria MUST include examinations of the child elements inside the SOAP body element  
 1344 `.../osci:MsgSelector.`

1345 Selected messages MUST be given back to the requestor one by one in the response to this request  
 1346 in an ascending order given by the values of the SOAP header block element  
 1347 `/osci:MsgTimeStamps/osci:Delivery` ("FIFO"). A `MsgBox` service MUST hold the complete  
 1348 list corresponding to the selection criteria and deliver an ID for this list to the requestor with the  
 1349 response. In subsequent requests (see `MsgBoxGetNextRequest` in chapter [8.2.4]) the requestor is  
 1350 able to pull further messages of a selection result with reference to this list. Remaining messages of  
 1351 the complete list MUST be retained for following messages of type `MsgBoxGetNextRequest`.

## 1352 8.2.2 MsgBoxStatusListRequest

1353 To request a message status list from a `MsgBox` service endpoint, a requestor MUST send a  
 1354 `MsgBoxStatusListRequest` message to his `MsgBox` instance endpoint.

1355 The normative outline for a `MsgBoxStatusListRequest` request is akin to the `MsgBoxFetchRequest`:

```

1356 <s12:Envelope ... >
1357   <s12:Header ... >
1358     ...
1359     <wsa:Action>
1360 http://www.osci.eu/ws/2008/05/transport/urn:messageTypes/MsgBoxStatusListRe
1361 quest
1362   </wsa:Action>
1363   <wsa:MessageID>xs:anyURI </wsa:MessageID>
1364   <wsa:To>xs:anyURI </wsa:To>
1365   <osci:TypeOfBusinessScenario wsa:ReferenceParameters="1">
1366     xs:anyURI
1367   </osci:TypeOfBusinessScenario>
1368   ...
1369 </s12:Header>
1370 <s12:Body ... >
1371   <osci:MsgBoxStatusListRequest maxListItems="xs:positiveInteger"
1372     ListForm=("MsgAttributes" | "MessageMetadata")?>
1373     <osci:MsgSelector newEntry=("true" | "false")>
1374       <osci:MessageID> xs:anyURI </osci:MessageID> *
1375       <osci:RelatesTo> xs:anyURI </osci:RelatesTo> *
1376       <osci:MsgBoxEntryTimeFrom>
1377         xs:dateTime
1378       </osci:MsgBoxEntryTimeFrom> ?
1379       <osci:MsgBoxEntryTimeTo>
1380         xs:dateTime
1381       </osci:MsgBoxEntryTimeTo> ?
1382       <osci:Extension> xs:anyType </osci:Extension> ?
1383     </osci:MsgSelector> ?
1384   </osci:MsgBoxStatusListRequest>
1385 </s12:Body>
1386 </s12:Envelope>
  
```

1388 Description of normative constraints on the outline listed above:

1389 `/s12:Envelope/s12:Header/wsa:Action`

1390 The value indicated herein MUST be used for that URI.

1391 `/s12:Envelope/s12:Header/wsa:MessageID`

1392 The request MUST carry a unique WS-Addressing MessageID.

1393 `/s12:Envelope/s12:Header/wsa:To`

1394 The address of the `MsgBox` (request destination) endpoint.

1395 `/s12:Envelope/s12:Header/osci:TypeOfBusinessScenario`

1396 This value of the instantiation of `/wsa:ReferenceParameters` MUST be supplied for this  
 1397 message type. The value of `/osci:TypeOfBusinessScenario` is taken as message  
 1398 selection argument and SHOULD match one of those accepted by this endpoint. As an  
 1399 alternative in this special case a value of "\*" MAY be supplied here, to select the message  
 1400 status list for all messages in this `MsgBox` instance. Such an entry MUST lead to a message

- 1401 box status list containing all messages, with no regard to a specific addressed business  
 1402 scenario of this endpoint, actually exposed as able to serve.
- 1403 Only status lists of messages originally targeted to the EPR outlined MUST be returned. If a  
 1404 `MsgBoxFetchRequest` contains no other arguments for message selection in the SOAP body  
 1405 element `osci:MsgBoxStatusListRequest`, the messages to be selected MUST be  
 1406 those which have not yet been fetched. That are all messages in the addressed `MsgBox`  
 1407 having no SOAP header element or the value zero in the  
 1408 `.../osci:MsgTimeStamps/osci:InitialFetched` time instant element.
- 1409 `/s12:Envelope/s12:Header/osci:TypeOfBusinessScenario/`  
 1410 `@wsa:IsReferenceParameter`
- 1411 In the following WS-Addressing, the element MUST be attributed with  
 1412 `@wsa:IsReferenceParameter="1"`
- 1413 The body of this message contains the actual request in the structure
- 1414 `/s12:Envelope/s12:Body/osci:MsgBoxStatusListRequest`
- 1415 Container holding detailed selection arguments in addition to  
 1416 `/s12:Envelope/s12:Header/osci:TypeOfBusinessScenario` above; this element  
 1417 MAY contain no child elements if no further selection criteria shall be provided.
- 1418 `/s12:Envelope/s12:Body/osci:MsgBoxStatusListRequest/@maxListItems ?`
- 1419 The requestor MAY limit the length of the message status list he expects in the response  
 1420 with this attribute of type `xs:positiveInteger`. A `MsgBox` service MUST hold the  
 1421 complete list corresponding to the selection criteria and deliver an ID for this list to the  
 1422 requestor together with the response. In subsequent requests (see `MsgBoxGetNextRequest`  
 1423 in chapter [8.2.4]), the requestor is able to request further portions of a selection result with  
 1424 reference to this list.
- 1425 A `MsgBox` instance MAY limit the value of `@maxListItems` to any value greater zero.
- 1426 If provided, a `MsgBox` instance MUST retain this value – if not decreased by its configured  
 1427 limit - together with the result set until the whole result set is delivered to the requestor or the  
 1428 requestor cancels an iteration sequence (see `MsgBoxCloseRequest` in chapter [8.2.5].
- 1429 `/s12:Envelope/s12:Body/osci:MsgBoxStatusListRequest/@ListForm ?`
- 1430 This optional attribute of type `xs:NMTOKEN` with the values listed below allows the recipient  
 1431 to denote whether the `osci:MsgStatusList` to be returned shall contain:
- 1432 • "MsgAttributes" –for selected messages it returns a sequence of  
 1433 `osci:MsgAttributes` elements as described in [8.2.3.2]. This is the default value  
 1434 and functionality as specified in version 2.0 of this specification
  - 1435 • "MessageMetaData": –for selected messages it returns a sequence of  
 1436 `oscimeta:MessageMetaData` elements as described in [8.2.3.2].
- 1437 **NOTE:** This attribute has been introduced with version 2.0.1 of this specification.  
 1438

1439 /s12:Envelope/s12:Body/osci:MsgBoxStatusListRequest/osci:MsgSelector

1440 For the content of this complex element, which defines selection criteria for messages, see  
1441 description in last chapter [8.2.1].

1442 /s12:Envelope/s12:Body/osci:MsgBoxStatusListRequest/osci:MsgSelector/  
1443 osci:Extension/{any} \*

1444 This is an extensibility mechanism to allow other search criteria to be passed. See respective  
1445 explanation for osci:MsgBoxStatusListRequest.

1446 Upon receipt and authentication of this message, the MsgBox service will locate any message that  
1447 matches the selection criteria. Only messages originally targeted to this EPR MUST be selected for  
1448 the required message status list. The search criteria MUST include examinations of the child elements  
1449 inside the /osci:MsgSelector SOAP body element.

1450 The message status list that is to be given back to the requestor, MUST be of the maximum size  
1451 denoted by /osci:MsgBoxStatusListRequest/@maxListItems or a lower size according to  
1452 possible configured restrictions of the requested MsgBox instance. The list MUST be built up and  
1453 sorted in an ascending order, given by the message SOAP header block element  
1454 /osci:MsgTimeStamps/osci:Delivery ("FIFO"). Remaining items of the complete list, not  
1455 deliverable to the requestor directly in the response to the initial MsgBoxStatusListRequest, MUST be  
1456 retained for following messages of type MsgBoxGetNextRequest.

### 1457 8.2.3 MsgBoxResponse

1458 Request messages MsgBoxFetchRequest und MsgBoxStatusListRequest are both responded by the  
1459 same status information in the SOAP header block of the response, only the body parts differ as  
1460 outlined in the following chapters.

1461 **NOTE:** It is strongly recommended to encrypt the SOAP body block of a MsgBoxResponse using the  
1462 Recipients X509 encryption certificate.

1463 The normative outline for a MsgBoxResponse header is:

```

1464 <s12:Envelope ... >
1465 <s12:Header ... >
1466 ...
1467 <wsa:Action>
1468   http://www.osci.eu/ws/2008/05/transport/urn/messageTypes/MsgBoxResponse
1469 </wsa:Action>
1470 <wsa:MessageID>xs:anyURI </wsa:MessageID>
1471 <wsa:FaultTo> wsa:EndpointReference </wsa:FaultTo> ?
1472 ...
1473 <osci:MsgBoxResponse MsgBoxRequestID="xs:anyURI"
1474   wsu:Id="xs:ID" ?
1475   <osci:NoMessageAvailable
1476     reason=
1477     ( "http://www.osci.eu/ws/2008/05/transport/MsgBox/reasons/NoMatch" |
1478     "http://www.osci.eu/ws/2008/05/transport/MsgBox/reasons/SearchArgumentInvalid" |
1479     "http://www.osci.eu/ws/2008/05/transport/MsgBox/reasons/RequestIDInvalid" |
1480     "xs:anyUri" />
1481     |
1482     <osci:ItemsPending>
1483       xs:positionInteger
1484     </osci:ItemsPending>
1485   </osci:NoMessageAvailable>
1486   ...
1487 </s12:Header>
1488 <s12:Body ... >
1489 ...
1490 </s12:Body>
1491 </s12:Envelope>

```

1492 Description of normative constraints of the outline listed above (WS-Addressing header elements are  
1493 to be handled according to chapter [6, Addressing Endpoints]):

1494 `/s12:Envelope/s12:Header/wsa:Action`

1495 The value indicated herein MUST be used for that URI.

1496 `/s12:Envelope/s12:Header/wsa:MessageID`

1497 The request MUST carry a unique WS-Addressing MessageID.

1498 `/s12:Envelope/s12:Header/wsa:FaultTo ?`

1499 The optional Endpoint Reference (EPR) SOAP fault messages should be routed to.

1500 `/s12:Envelope/s12:Header/osci:MsgBoxResponse`

1501 The container for the status response; the status response is a choice of two alternatives,  
1502 depending on request fulfilment.

1503 The following attributes MUST be set:

1504 `/s12:Envelope/s12:Header/osci:MsgBoxResponse/@MsgBoxRequestID`

1505 This mandatory element of type `xs:anyURI` MUST carry a unique value of type UUID  
1506 according to [RFC4122]. It serves to identify messages of type `MsgBoxFetchRequest` and  
1507 `MsgBoxStatusListRequest` and MUST be used by the requestor in subsequent messages of  
1508 type `MsgBoxGetNextRequest` or `MsgBoxCloseRequest` explained below. The value MUST  
1509 be generated by a `MsgBox` instance for every incoming `MsgBoxFetchRequest` or  
1510 `MsgBoxStatusListRequest` and MUST be retained and correlated to these requests with their  
1511 individual search criteria.

1512 `/s12:Envelope/s12:Header/osci:MsgBoxResponse/@wsu:Id ?`

1513 For ease of referencing this SOAP body block, this optional attribute of type `wsu:Id` MAY be  
1514 provided.

1515 `/s12:Envelope/s12:Header/osci:MsgBoxResponse/osci:NoMessageAvailable`

1516 This element of the choice of `.../MsgBoxResponse` MUST be set if

1517 there are no messages available. Corresponding to the selection criteria

1518 there where errors detected in the selection criteria.

1519 The element carries the following attribute:

1520 `/s12:Envelope/s12:Header/osci:MsgBoxResponse/osci:NoMessageAvailable/`

1521 `@reason`

1522 Attribute of type `xs:anyURI`, identifies the reason of `/osci:NoMessageAvailable` set  
1523 with the following defined meanings:

<b>@reason URI</b>	<b>Meaning</b>
<code>http://www.osci.eu/ws/2008/05/transport/MsgBox/reasons/NoMatch</code>	No messages matching the search criteria could be found
<code>http://www.osci.eu/ws/2008/05/transport/MsgBox/reasons/SearchArgsInvalid</code>	Error contained in search arguments
<code>http://www.osci.eu/ws/2008/05/common/urn/MsgBox/reasons/RequestIdInvalid</code>	RequestId of subsequent GetNext- Close- Request is not known or no longer available at the <code>MsgBox</code> instance

@reason URI	Meaning
Any other URI	Specific other reasons MAY be defined by implementations

1524 Table 8: Predefined business scenario types

1525 The alternative of the choice of `.../MsgBoxResponse` MUST be set if there are – according to the  
 1526 selection criteria of the request - messages or message status list items pending (not yet deliverable  
 1527 to the requestor in the actual response):

1528 `/s12:Envelope/s12:Header/osci:MsgBoxResponse/osci:ItemsPending`

1529 This element of type `xs:nonNegativeInteger` MUST be set with the number of the  
 1530 remaining items. If this is the last portion of a result set delivered to the requestor with this  
 1531 actual response, the value MUST be set to zero to signal this fact.

### 1532 8.2.3.1 *MsgBoxResponse - body to MsgBoxFetchRequest*

1533 For this type of foregoing initial request, the requested message MUST be delivered in following the  
 1534 manner:

#### 1535 **Rule for OSCI version 2.0 conformant implementations:**

- 1536 • A SOAP Envelope with all child elements MUST be build up containing a header block with  
 1537 the ones of the original message, except for header elements, which have initially been  
 1538 targeted to and successfully executed by the MsgBox node, as well as the transport  
 1539 encryption and signature elements, which have been supplied by the Initiator of this message.
- 1540 • The SOAP header element `osci:MsgTimeStamps` MUST be inserted (or completed, if  
 1541 present) as described in chapter [8.1].
- 1542 • All original WS-Addressing, `osci:X509TokenContainer`, `xkms:ValidateResult` (inside  
 1543 a `xkms:CompoundResult`) and original security token and WS-Trust header elements MUST  
 1544 be included.
- 1545 • If present in the original message, the `osci:ReceptionReceiptDemand` header element  
 1546 MUST be included.
- 1547 • If present in the original message, the `oscimeta:MessageMetaData` header element MUST  
 1548 be included.
- 1549 • The original SOAP body child elements MUST be included unchanged as child elements of  
 1550 the SOAP body of this SOAP Envelope to be built up.
- 1551 • The recipient may have interest in the authentication and authorization data originally carried  
 1552 in the SOAP WS Security header when delivering a message to the recipient's MsgBox.  
 1553 Therefore, this security token MUST be inserted as additional child element into the SOAP  
 1554 header of this SOAP Envelope to be built up.

1555 The resulting SOAP envelope MUST be included as child element of the SOAP body block of the  
 1556 response message.

#### 1557 **Rule for implementations conformant to OSCI version 2.0.1 and higher:**

1558 **If the value of attribute @MsgPart of `osci:MsgBoxFetchRequest` is set to**

- 1559 • **"Envelope": rule above applies – whole resulting SOAP envelope MUST be included**  
 1560 **as child element of the SOAP body block of the response message.**
- 1561 • **"Header": whole resulting SOAP header elements MUST be included as child**  
 1562 **element of the SOAP body block of the response message.**

- 1563                   • "Body": only the original SOAP body child element<sup>21</sup> MUST be included unchanged as  
1564                   child element in the SOAP body of the response message.

### 1565 **8.2.3.2 MsgBoxResponse - Body to MsgBoxStatusListRequest**

1566 For this type of foregoing initial request, the requested list MUST be built up in the SOAP body block  
1567 of the response message. This is the same for responses to subsequent requests of type  
1568 MsgBoxGetNextRequest (see MsgBoxGetNextRequest in chapter [8.2.4]).

1569 The normative outline for a MsgStatusList is:

```
1570 <osci : MsgSt at usLi st >
1571   <osci : MsgAt t i but es>
1572     <wsa: MessageID>xs: anyUri </ wsa: MessageID>
1573     <wsa: Rel at esTo>xs: anyUri </ wsa: Rel at esTo> *
1574     <wsa: Fr om>endpoi nt - r ef er ence</ wsa: Fr om> ?
1575     <osci : TypeOf Busi nessScenar i o>xs: anyUri </ osci : TypeOf Busi nessScenar i o>
1576     <osci : MsgSi ze>xs: posi t i vel nt eger </ osci : msgSi ze>
1577     <osci : Obsol et eAf t er Dat e> xs: dat e </ osci : Obsol et eAf t er Dat e> ?
1578     <osci : Del i veryTi me> xs: dat eTi me </ osci : Del i veryTi me>
1579     <osci : I ni t i al Fet chedTi me> xs: dat eTi me </ osci : I ni t i al Fet chedTi me> ?
1580   </ osci : MsgAt t r i but es> *
1581   </ osci met a: MessageMet aDat a/ > *
1582 </ osci : MsgSt at usLi st >
```

1583 The whole structure MUST be positioned under `/s12:Envelope/s12:Body`.

1584 `/osci:MsgStatusList`

1585                   Container for the items of the list. According to the value of attribute `@ListForm` of the  
1586                   foregoing `osci:MsgBoxStatusListRequest`, the list MUST be built of a sequence of  
1587                   `../osci:MsgAttributes` (default behaviour) `../oscimeta:MessageMetaData`  
1588                   elements.<sup>22</sup>

1589                   For selected messages not carrying a header `/oscimeta:MessageMetaData`,  
1590                   `../osci:MsgAttributes` MUST be returned.

1591 `/osci:MsgStatusList/osci:MsgAttributes *`

1592                   The container for the attributes of one message of the status list. The number of occurrences  
1593                   is determined by the number of items of the selection result list not yet delivered to the  
1594                   requestor and the value of `../osci:MsgBoxStatusListRequest/@maxListItems` of  
1595                   the initial `MsgBoxStatusListRequest`, which MAY be modified to a lower value greater zero  
1596                   set by the requested `MsgBox` instance.

1597 `/osci:MsgStatusList/osci:MsgAttributes/wsa:MessageID`

1598                   MessageID of the message, derived from respective header element.

1599 `/osci:MsgStatusList/osci:MsgAttributes/wsa:RelatesTo *`

1600                   MessageIDs of related messages of the message, derived from respective header elements,  
1601                   if present there they MUST be included in `/osci:MsgStatusList`.

1602 `/osci:MsgStatusList/osci:MsgAttributes/wsa:From ?`

1603                   Optional element, From-EPR of the message, derived from the respective header element, if  
1604                   present there it MUST be included in `/osci:MsgStatusList`.

1605 `/osci:MsgStatusList/osci:MsgAttributes/osci:TypeOfBusinessScenario`

<sup>21</sup> As recommended in chapter [5], a SOAP body is assumed to carry only one child element.

<sup>22</sup> Due to downwards compatibility, `/osci21:MessageMetaData`, `../osci:MsgAttributes` are not defined as choice.

1606 This URI denotes the type of addressed business scenario of the intended recipient of the  
 1607 message. It is derived from the `/wsa:ReferenceParameters`  
 1608 `/osci:TypeOfBusinessScenario` associated to the WS-Addressing SOAP header  
 1609 element `/wsa:To` of the message.

1610 `/osci:MsgStatusList/osci:MsgAttributes/osci:MsgSize`

1611 The size of the message in kilobytes, has to be supplied here as `xs:positiveInteger`.

1612 The following timestamps are provided in an `osci:MsgStatusList` according to the  
 1613 `/osci:MsgTimeStamps` described in chapter [8.1]:

1614 `/osci:MsgStatusList/osci:MsgAttributes/ObsoleteAfterDate ?`

1615 Optional element of type `xs:date`, contains - if present in the underlying message - the  
 1616 value of the SOAP header block element  
 1617 `/osci:MsgTimeStamps/osci:ObsoleteAfter` present in the underlying message.

1618 `/osci:MsgStatusList/osci:MsgAttributes/DeliveryTime`

1619 This element of type `xs:dateTime` contains the value of the SOAP header block element  
 1620 `.../osci:MsgTimeStamps/osci:Delivery`, which MUST be present in a message  
 1621 stored by a `MsgBox` instance.

1622 `/osci:MsgStatusList/osci:MsgAttributes/InitialFetchTime ?`

1623 This optional element of type `xs:dateTime` contains the value of the SOAP header block  
 1624 element `.../osci:MsgTimeStamps/osci:InitialFetch`, which MAY be present in a  
 1625 message stored by a `MsgBox` instance. Only if not present or present with the value zero,  
 1626 the `MsgBox` instance MUST provide this element with its actual server time before message  
 1627 delivery.

1628 `/osci:MsgStatusList/oscimeta:MessageMetaData *`

1629 Sequence of according header block elements of selected messages. For number of  
 1630 sequence entries see `.../MsgAttributes` above.

## 1631 8.2.4 MsgBoxGetNextRequest

1632 To request subsequent, not yet delivered results from foregoing requests of type  
 1633 `MsgBoxStatusListRequest` or `MsgBoxFetchRequest`, a requestor MUST send a  
 1634 `MsgBoxGetNextRequest` message to the same `MsgBox` instance.

1635 The normative outline for a `MsgBoxGetNextRequest`:

```

1636 <s12:Envelope ... >
1637   <s12:Header ... >
1638     ...
1639     <wsa:Action>
1640       http://www.osci.eu/ws/2008/05/transport/urn/messageTypes/
1641       MsgBoxGetNextRequest
1642     </wsa:Action>
1643     <wsa:MessageID>xs:anyURI </wsa:MessageID>
1644     <wsa:To>xs:anyURI </wsa:To>
1645     <osci:TypeOfBusinessScenario wsa:IsReferenceParameter="1">
1646       xs:anyURI </osci:TypeOfBusinessScenario>
1647     ...
1648   </s12:Header>
1649   <s12:Body ... >
1650     <osci:MsgBoxGetNextRequest MsgBoxRequestID="xs:anyURI">
1651       <osci:LastMsgReceived wsa:MessageID </osci:LastMsgReceived> *
1652     </osci:MsgBoxGetNextRequest>
1653   </s12:Body>
1654 </s12:Envelope>
  
```

1655 Description of normative constraints on the outline listed above:

1656 `/s12:Envelope/s12:Header/wsa:Action`

- 1657 The value indicated herein MUST be used for that URI.
- 1658 `/s12:Envelope/s12:Header/wsa:MessageID`
- 1659 The request MUST carry a unique WS-Addressing MessageID.
- 1660 `/s12:Envelope/s12:Header/wsa:To`
- 1661 The address of the MsgBox (request destination) endpoint.
- 1662 `/s12:Envelope/s12:Header/osci:TypeOfBusinessScenario`
- 1663 The corresponding value of the initial MsgBoxFetchRequest or MsgBoxStatusListRequest
- 1664 MUST be supplied for this message type.
- 1665 `/s12:Envelope/s12:Header/osci:TypeOfBusinessScenario/`
- 1666 `@wsa:IsReferenceParameter`
- 1667 According to WS-Addressing, the element MUST be attributed with
- 1668 `@wsa:IsReferenceParameter="1"`
- 1669 The body of this message contains the actual
- 1670 `/s12:Envelope/s12:Body/osci:MsgBoxGetNextRequest.`
- 1671 `/s12:Envelope/s12:Body/osci:MsgBoxGetNextRequest/@MsgBoxRequestID`
- 1672 This attribute of type `xs:anyURI` MUST be provided with the value of the foregoing
- 1673 `MsgBoxResponse/@MsgBoxRequestID`. The MsgBox service MUST use it to correlate
- 1674 this MsgBoxGetNextRequest to the initial MsgBoxFetchRequest respective
- 1675 MsgBoxStatusListRequest.
- 1676 `/s12:Envelope/s12:Body/osci:MsgBoxGetNextRequest/osci:LastMsgReceived *`
- 1677 These optional elements of type `wsa:AttributedURIType` MAY be provided, when the
- 1678 underlying initial request was of type MsgBoxFetchRequest. The requestor SHOULD provide
- 1679 here the value(s) of the `/wsa:MessageID` of the last message(s) he received in the body of
- 1680 the foregoing response(s) to commit successful reception of those messages. This has to be
- 1681 realized as "reception acknowledged by requester" by the MsgBox instance: If the SOAP
- 1682 header element `.../osci:MsgTimeStamps/osci:Reception` is absent or the value of the
- 1683 SOAP header element `.../osci:MsgTimeStamps/osci:Reception` is zero, the actual
- 1684 server time of the MsgBox instance MUST now be set here and the value has to be signed
- 1685 according to chapter [8.1]. The resulting changes in the SOAP header block
- 1686 `/osci:MsgTimeStamps` MUST now be persisted in the MsgBox store.
- 1687 Upon receipt and authentication of this message, the MsgBox service MUST – depending on type of
- 1688 initial request referenced by `osci:MsgBoxGetNextRequest/@MsBoxRequestID`
- 1689 – deliver a MsgBoxResponse with the next message of the list indicated by
  - 1690 `/osci:MsgBoxResponse/@MsBoxRequestID` in the body of the response (rules denoted
  - 1691 in chapter [8.2.3.1] apply)
  - 1692 – deliver a MsgBoxResponse with the next portion of a `/osci:MsgStatusList` indicated by
  - 1693 `/osci:MsgBoxResponse/@MsBoxRequestID` in the body of the response (rules denoted
  - 1694 in chapter [8.2.3.1] apply).
- 1695 Inside the SOAP header the element `.../osci:MsgBoxResponse`, choice `.../osci:ItemsPending`
- 1696 MUST be set to the actual value. If `.../osci:ItemsPending` becomes the value zero, this fact
- 1697 signals the requestor that the MsgBox instance may have discarded the search result list referenced
- 1698 by the identifier `/osci:MsgBoxResponse/@MsBoxRequestID`.

## 1699 8.2.5 MsgBoxCloseRequest

1700 The functionalities of this message type are:

- 1701           • Recipient successful commits reception of messages from his MsgBox instance
- 1702           • Recipient signals the abortion of an iterative pull process of sequences of result requests of
- 1703           foregoing initial MsgBoxFetchRequest respective MsgBoxStatusListRequest.

1704 **NOTE:** In case of successful processing of a MsgBoxCloseRequest by the targeted MsgBox instance  
1705 a response MUST NOT be generated.

1706 The normative outline for the MsgBoxCloseRequest:

```

1707 <s12:Envelope ... >
1708   <s12:Header ... >
1709     ...
1710     <wsa:Action>
1711     http://www.osci.eu/ws/2008/05/transport/urn:messageTypes/MsgBoxCloseRequest
1712     </wsa:Action>
1713     <wsa:MessageID>xs:anyURI </wsa:MessageID>
1714     <wsa:To>xs:anyURI </wsa:To>
1715     <osci:TypeOfBusinessScenario wsa:IsReferenceParameter="1">
1716     xs:anyURI
1717     </osci:TypeOfBusinessScenario>
1718     ...
1719     </s12:Header>
1720     <s12:Body ... >
1721     <osci:MsgBoxCloseRequest MsgBoxRequestID="xs:anyURI">
1722     <osci:LastMsgReceived>wsa:MessageID</LastMsgReceived> *
1723     </osci:MsgBoxCloseRequest>
1724     </s12:Body>
1725   </s12:Envelope>
1726

```

1727 Description of normative constraints on the outline listed above:

1728 **/s12:Envelope/s12:Header/wsa:Action**

1729           The value indicated herein MUST be used for that URI.

1730 **/s12:Envelope/s12:Header/wsa:MessageID**

1731           The request MUST carry a unique WS-Addressing MessageID.

1732 **/s12:Envelope/s12:Header/wsa:To**

1733           The address of the MsgBox (request destination) endpoint.

1734 **/s12:Envelope/s12:Header/osci:TypeOfBusinessScenario**

1735           The corresponding value of the initial MsgBoxFetchRequest or MsgBoxStatusListRequest  
1736           MUST be supplied for this message type.

1737 **/s12:Envelope/s12:Header/osci:TypeOfBusinessScenario/**

1738 **@wsa:IsReferenceParameter**

1739           According to WS-Addressing, the element MUST be attributed with

1740 **@wsa:IsReferenceParameter="1"**

1741 The body of this message contains the actual

1742 **/s12:Envelope/s12:Body/osci:MsgBoxCloseRequest.**

1743 **/s12:Envelope/s12:Body/osci:MsgBoxGetNextRequest/@MsgBoxRequestID**

1744           This attribute of type **xs:anyURI** MUST be provided with the value of the foregoing  
1745 **MsgBoxResponse/@MsgBoxRequestID**. The MsgBox service MUST use it to correlate  
1746 this MsgBoxCloseRequest to the initial MsgBoxFetchRequest respective  
1747 MsgBoxStatusListRequest.

1748 **/s12:Envelope/s12:Body/osci:MsgBoxCloseRequest/LastMsgReceived ?**

1749           These optional elements of type **wsa:AttributedURIType** MAY be provided, when the  
1750           underlying initial request was of type MsgBoxFetchRequest. The requestor SHOULD provide

1751 here the value(s) of the `/wsa:MessageID` of the last message(s) he received in the body of  
 1752 the foregoing response(s) to successfully commit reception of those messages. This has to  
 1753 be realized as "reception acknowledged by requester" by the `MsgBox` instance: If the SOAP  
 1754 header element `.../osci:MsgTimeStamps/osci:InitialFetch` is absent or present  
 1755 with the value zero, the actual server time of the `MsgBox` instance MUST now be set here  
 1756 and the value has to be signed according to chapter [8.1]. The resulting changes in the  
 1757 SOAP header block `/osci:MsgTimeStamps` MUST now be persisted in the `MsgBox` store.

1758 It should be noted, that this message type MUST be sent to the `MsgBox` instance always  
 1759 when a `MsgBoxFetchRequest` was the initial message sent and the requestor pulled a  
 1760 message successfully the first time (a new message). This triggers the commitment of the  
 1761 SOAP header `/osci:MsgTimeStamps/osci:Reception` and  
 1762 `/osci:MsgTimeStamps/osci:InitialFetch` time instances. *It is up to*  
 1763 *implementations of recipient instances, how to distinguish between "new" and already*  
 1764 *processed messages, because the recipient transport gateway has no implicit control on the*  
 1765 *state of the successful body processing (for example to mark message as "read" or*  
 1766 *"processed" – this at least is under the control of the application targeted by the message).*

1767 To avoid situations, where successfully pulled messages on the `MsgBox` instance side  
 1768 remain in the state unpulled, it is strongly recommended to commit every `MsgBoxResponse`  
 1769 to an initial `MsgBoxFetchRequest` and following series of `MsgBoxGetNextRequest`.

## 1770 8.2.6 Processing Rules for `MsgBoxGetNext/CloseRequest`

1771 `MsgBox` instances are free to configure a timeout value to retain search result lists identified by  
 1772 `/osci:MsgBoxResponse/@MsgBoxRequestID`.

1773 If a `MsgBox` instance receives a `MsgBoxGetNextRequest` or a `MsgBoxCloseRequest` not at all or no  
 1774 longer known here, no processing on the message database must be done and a following fault  
 1775 MUST be generated:

1776 Fault 9: **`MsgBoxRequestWrongReference`**

1777 [Code] Sender

1778 [Subcode] `MsgBoxRequestWrongReference`

1779 [Reason] `MsgBoxRequestID` unknown or timed out.

## 1780 8.3 Receipts

1781 Requirements for receipting message exchange were outlined in "OSCI-Transport 2.0 – Functional  
 1782 Requirements and Design Objectives" and "OSCI-Transport 2 – Technical Features Overview"

1783 Besides provableness of what has been delivered / received when, for messages exchange patterns  
 1784 using the `MsgBox` service it may be of interest for the Initiator to be informed, when the intended  
 1785 recipient pulls the message from his `MsgBox`. More concrete – the business-scenario-needs of an  
 1786 asynchronous message are bound to reaction times. In this case, a service requestor has to have  
 1787 control to in-time delivery to the targeted recipient. In doubt of recipient activity concerning the request,  
 1788 a service requestor (or even responder) may choose other communication channels to get in contact.

1789 As there may be non-conformant implementations which don't answer to a requested  
 1790 `ReceptionReceipt`, for additional comfort of control whether a message has been pulled yet by the  
 1791 intended recipient, the construct of a **FetchNotification** is foreseen, which alike described for  
 1792 receipts, can be demanded by initiator and recipient instances. If requested, the recipients `MsgBox`  
 1793 instance MUST deliver such a notification to the endpoint, the initiator specified in his message;  
 1794 contents are the SOAP header elements indicating source and destination of the message and the  
 1795 time instant when it is pulled by the intended recipient. No separate signature is foreseen for this  
 1796 notification. The `FetchNotification` is delivered in the SOAP body of a separate `osci:Request` to the

1797 endpoint to be exposed in the demand for this FetchedNotification – which again in general should be  
1798 the MsgBox of the requesting initiator or recipient node.

### 1799 **8.3.1 Demanding Receipts**

1800 To demand receipts and define its details, for each specific demand the here defined SOAP header  
1801 blocks MAY be provided in outbound messages of type osci:Request and osci:Response by  
1802 SOAP/OSCI endpoints (initiator or recipient).

#### 1803 **8.3.1.1 Demand for Delivery Receipt**

1804 If at the next logical OSCI node a message of type osci:Request is targeted is requested to deliver a  
1805 DeliveryReceipt in the backchannel of the osci:Response message, the following SOAP header block  
1806 MUST be included in the message:

```
1807 <osci:DeliveryReceiptDemand wsu:Id="xs:ID"
1808   @s12:role=
1809   "http://www.w3.org/2003/05/soap-envelope/role/next" ?
1810   @s12:mustUnderstand="true" | "false" ?
1811   @qualTSPforReceipt="true" | "false" ?
1812   @echoRequest="true" | "false" ? >
1813   <wsa:ReplyTo> wsa:EndpointReference </wsa:ReplyTo>
1814 </osci:DeliveryReceiptDemand> ?
```

1815 Description of elements and attributes in the schema overview above:

1816 **/osci:DeliveryReceiptDemand ?**

1817 Optional SOAP header for indicating requirements for a DeliveryReceipt. It MUST be  
1818 provided under the conditions mentioned above.

1819 **/osci:DeliveryReceiptDemand/@wsu:Id**

1820 This attribute of type **wsu:Id** SHOULD be provided so that unambiguous references can be  
1821 applied to this element.

1822 **/osci:DeliveryReceiptDemand/@s12:role**

1823 This attribute of type **xs:anyURI** MAY be provided. It defaults to the URI outlined above. If  
1824 this attribute is provided, it MUST be set to this value. According to the semantics of  
1825 [SOAP12], this SOAP header block is designated to the next SOAP-node passed on the  
1826 message route.

1827 **/osci:DeliveryReceiptDemand/@s12:mustUnderstand**

1828 This Boolean attribute SHOULD be provided with the value "true". Following the semantics  
1829 of [SOAP12], this SOAP header block MUST be understood and processed by the next  
1830 SOAP-node passed on the message route willing to act in the role denoted by the foregoing  
1831 attribute **/osci:ReceiptDemand/@s12:role**. For interoperability reasons with web  
1832 service implementations not able to process the receipts defined here, it may be set to  
1833 "false" or not present (which is equivalent to the value "false").

1834 **/osci:DeliveryReceiptDemand/@qualTSPforReceipt ?**

1835 This optional Boolean attribute signals – if set to "true" – a qualified timestamp for the  
1836 requested receipt information. If such a service is not available on the node, the receipt is  
1837 demanded from, a fault (see chapter [8.3.2]) MUST be generated to the requesting node and  
1838 the incoming message MUST be discarded.

1839 **/osci:DeliveryReceiptDemand/@echoRequest ?**

1840 This optional Boolean attribute signals – if set to "true" – that the requesting node requires  
1841 the retransmission of the whole message in the required receipt. In this case, the node the  
1842 receipt is demanded from, MUST provide the whole message in binary format in the receipt

1843 part of the response message (see chapter [8.3.2.1]). Care should be taken to use this  
1844 feature with regard to caused overhead and bandwidth consumption.

1845 If absent, this attribute defaults to "false".

1846 **/osci:DeliveryReceiptDemand/wsa:ReplyTo**

1847 This required element of type **wsa:EndpointReferenceType** denotes the endpoint,  
1848 where the requestor wishes the receipt should be routed to. In case of a DeliveryReceipt  
1849 demand in a message of type **osci:Request**, the value herein for **.../wsa:Address** SHOULD  
1850 be **http://www.w3.org/2005/08/addressing/anonymous**; the DeliveryReceipt is  
1851 returned directly in the header of the response to the incoming message in the same HTTP-  
1852 connection.

1853 If the requestor wishes that a DeliveryReceipt should be routed, some specialized endpoint  
1854 consuming receipts of the EPR of the endpoint MUST be exposed here. The DeliveryReceipt  
1855 in this case MUST be delivered in the SOAP body of a separate new **osci:Request** message.  
1856 Hence, this EPR SHOULD be the one of the **MsgBox** instance of the requestor. It MAY even  
1857 be a specialized endpoint consuming receipt. The EPR MUST contain reference properties  
1858 according to chapter [6, Addressing Endpoints]. A **.../wsa:ReferenceParameters** of  
1859 following value SHOULD be provided:

1860 **<osci:TypeOfBusinessScenario>**

1861 **www.osci.eu/ws/2008/05/common/urn/messageTypes/Receipt**

1862 **</osci:TypeOfBusinessScenario>**.

1863 For delivering a receipt to a **MsgBox** instance, this is the default value for separating receipt  
1864 message types from other ones (see chapter [6, Addressing Endpoints]).

1865 An **/osci:DeliveryReceiptDemand** header block MUST NOT be included in an **osci:Response**  
1866 message. As for an **osci:Response**, there is no network backchannel available; in this case  
1867 DeliveryReceipt could not be delivered in the standard manner. If provability of response delivery is  
1868 needed, an **/osci:ReceptionReceiptDemand** should be used instead.

1869 For synchronous request-response scenarios driven point-to-point between instances of initiator and  
1870 recipient, it is advisable to economize demands for receipts to avoid overhead and processing of not  
1871 needed DeliveryReceipts. Provability of communication here MAY be gained by a reception receipt  
1872 requirement, positioned in one or more messages of the request-response sequence, depending on  
1873 underlying concrete business process needs. Certainty of delivery itself is implicitly given by  
1874 successful processing of such a scenario.

### 1875 **8.3.1.2 Demand for ReceptionReceipt**

1876 If an endpoint a message of type **osci:Request** or **osci:Response** is targeted to shall deliver a  
1877 ReceptionReceipt, the following SOAP header block MUST be included in the message. The  
1878 underlying schema is the same as for an **osci:DeliveryReceiptDemand** SOAP header block;  
1879 possible attribute/element values and semantics differ in detail as described below.

1880

```

1881 <osci:ReceptionReceiptDemand wsu:Id="..."
1882   @s12:role=
1883   "http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver" ?
1884   @s12:mustUnderstand="true" | "false" ?
1885   @qualTSPforReceipt="true" | "false" ?
1886   @echoRequest="true" | "false") ? >
1887   <wsa:ReplyTo> wsa:EndpointReference <wsa:ReplyTo>
1888 </osci:ReceptionReceiptDemand ?

```

1889 Description of elements and attributes in the schema overview above:

1890 **/osci:ReceptionReceiptDemand ?**

1891 Optional SOAP header for indicating requirements for a ReceptionReceipt.

1892 **/osci:ReceptionReceiptDemand/@wsu:Id**

1893 This attribute of type **wsu:Id** SHOULD be provided so that unambiguous references can be  
1894 made to this element.

1895 **/osci:ReceptionReceiptDemand/@s12:role**

1896 This attribute of type **xs:anyURI** MAY be provided. It defaults to the URI outlined above. If  
1897 this attribute is provided, it MUST be set to this value; following the semantics of [SOAP12],  
1898 this SOAP header block is designated to a SOAP-node acting in the role of an ultimate  
1899 receiver – which is the node the SOAP body is finally designated to, corresponding to the  
1900 UltimateRecipient node in the role model of this specification.

1901 **/osci:ReceptionReceiptDemand/@s12:mustUnderstand**

1902 This Boolean attribute SHOULD be provided with the value "true". Following the semantics  
1903 of [SOAP12], this SOAP header block MUST be understood and processed by the next  
1904 SOAP-node, passed on the message route willing to act in the role denoted by the foregoing  
1905 attribute **/osci:ReceptionReceiptDemand/@s12:role**. For interoperability reasons, with web  
1906 service implementations not able to process the receipts defined here, it may be set to  
1907 "false" or not present (which is equivalent to "false").

1908 **/osci:ReceptionReceiptDemand/@qualTSPforReceipt ?**

1909 This optional Boolean attribute signals – if set to the value "true" – that a qualified timestamp  
1910 is requested for the receipt information. If such a service is not available on the node, the  
1911 receipt is demanded from, a fault (see chapter [8.3.2]) MUST be generated to the requesting  
1912 node and the message MUST be discarded.

1913 **/osci:ReceptionReceiptDemand/@echoRequest ?**

1914 This optional Boolean attribute signals – if set to the value "true" – that the requesting node  
1915 requires the retransmission of the whole message in the required receipt. In this case, the  
1916 node the receipt is demanded from MUST provide the whole message in binary format in the  
1917 receipt part of the response message (see chapter [8.3.2.2]). Care should be taken while  
1918 using this feature since it can cause overhead and bandwidth consumption.

1919 If absent, this attribute defaults to a value of "false".

1920 **/osci:ReceptionReceiptTo/wsa:ReplyTo**

1921 This required element of type **wsa:EndpointReferenceType** denotes the endpoint,  
1922 where the requestor wishes the receipt should be routed to. A ReceptionReceipt in general  
1923 MUST be delivered in the SOAP body of a separate new **osci:Request** message, hence this  
1924 EPR SHOULD be the one of the **MsgBox** instance of the requestor or MAY be a specialized  
1925 endpoint consuming receipts. The EPR MUST contain reference properties according to  
1926 chapter [6, Addressing Endpoints]. A **.../wsa:ReferenceParameters** of following value  
1927 SHOULD be provided:

1928 **<osci:TypeOfBusinessScenario>**

1929 `www.osci.eu/ws/2008/05/common/urn/messageTypes/Receipt`  
 1930 `</osci:TypeOfBusinessScenario>`.  
 1931 In case of delivering a receipt to a MsgBox instance, this is the default value for separating  
 1932 receipt message types from other ones (see chapter [6, Addressing Endpoints]).

### 1933 8.3.2 Receipt Format and Processing

1934 **NOTE:** To facilitate interoperable implementations, it is strongly recommended to use  
 1935 `"http://www.w3.org/2001/04/xmlenc#sha256"` as digest algorithm for the receipt signatures.

#### 1936 8.3.2.1 Delivery Receipt

1937 DeliveryReceipts MUST be produced immediately after successful acceptance of an incoming  
 1938 message of type osci:Request, if a SOAP header element `/osci:DeliveryReceiptDemand` is  
 1939 present in the incoming osci:Request message.

1940 The data for this type of receipt has to be carried in the resulting SOAP response message in the  
 1941 following SOAP header block `/osci:DeliveryReceipt`:

```

1942 <osci:DeliveryReceipt @wsu:Id="xs:ID"
1943 <osci:ReceiptInfo
1944   @wsu:Id="..."
1945   @osci:ReceiptIssuerRole=
1946     "http://www.osci.eu/ws/2008/05/transport/role/MsgBox" |
1947     "http://www.osci.eu/ws/2008/05/transport/role/Recipient" |
1948     "http://www.osci.eu/ws/2008/05/transport/role/Sender" |
1949     "http://www.osci.eu/ws/2008/05/transport/role/Relay" >
1950   <wsa:MessageD>xs:anyURI </wsa:MessageD>
1951   <osci:MsgTimestamp/>
1952   <wsa:RelatesTo/> *
1953   <osci:To> wsa:EndpointReference </osci:To>
1954   <wsa:ReplyTo> wsa:EndpointReference </wsa:ReplyTo>
1955   <wsa:From> wsa:EndpointReference </wsa:From> ?
1956   <osci:RequestEcho> xs:base64Binary </RequestEcho> ?
1957 </osci:ReceiptInfo>
1958 <ds:Signature/>
1959 </osci:DeliveryReceipt>
  
```

1960 Description of elements and attributes in the schema overview above:

1961 `/osci:DeliveryReceipt`

1962 Container holding the child elements receipt data `.../osci:ReceiptInfo` and a  
 1963 `ds:Signature` element over `.../osci:ReceiptInfo`.

1964 `/osci:DeliveryReceipt/@wsu:Id`

1965 This attribute of type `xs:ID` MUST be provided so that unambiguous references (i.e. for  
 1966 transport signature and encryption) can be made to this `/osci:DeliveryReceipt` block.

1967 `/osci:DeliveryReceipt/osci:ReceiptInfo`

1968 Container to hold the receipt details.

1969 `/osci:DeliveryReceipt/osci:ReceiptInfo/wsu:@Id`

1970 This attribute of type `xs:ID` MUST be provided; the element must be referenceable from the  
 1971 signature element described below.

1972

- 1973 **/osci:DeliveryReceipt/osci:ReceiptInfo/osci:ReceiptIssuerRole**
- 1974 This element of type **xs:anyURI** MUST be provided with one of the URIs outlined above.
- 1975 The concrete value MUST expose the role of the receipt issuing node. If an **osci:Request** is
- 1976 targeted to a **MsgBox** instance, the value MUST be
- 1977 "**http://www.osci.eu/ws/2008/05/transport/role/MsgBox**". If an **osci:Request**
- 1978 is targeted directly to the recipients OSCI Gateway or an **osci:Response** message containing
- 1979 a demand for a **DeliveryReceipt**, the value MUST be
- 1980 "**http://www.osci.eu/ws/2008/05/transport/role/Recipient**".
- 1981 If the receipt issuing node is a sender initially submitting the message, the value MUST be
- 1982 "**http://www.osci.eu/ws/2008/05/transport/role/Sender**"; if an active
- 1983 intermediary is just enriching and relaying the message, the value MUST be
- 1984 "**http://www.osci.eu/ws/2008/05/transport/role/Relay**".
- 1985 **/osci:DeliveryReceipt/osci:ReceiptInfo/wsa:MessageID**
- 1986 The **/wsa:MessageID** SOAP header block of the message to be receipted.
- 1987 **/osci:ReceptionReceipt/osci:ReceiptInfo/osci:MsgTimeStamps**
- 1988 The **/osci:MsgTimeStamps** SOAP header block; this element MUST be inserted after the
- 1989 receipting node has inserted its specific timestamps according to chapter [8.1].
- 1990 **/osci:DeliveryReceipt/osci:ReceiptInfo/wsa:RelatesTo \***
- 1991 The **/wsa:RelatesTo** SOAP header blocks of the message to be receipted.
- 1992 **/osci:DeliveryReceipt/osci:ReceiptInfo/osci:To**
- 1993 This element of type **wsa:EndpointReference** denotes the destination EPR of the
- 1994 message to be receipted. At least, it MUST contain the **/wsa:To** SOAP header block of this
- 1995 message and those SOAP header blocks attributed by
- 1996 **@wsa:IsReferenceParameter="1"**.
- 1997 **/osci:DeliveryReceipt/osci:ReceiptInfo/wsa:From ?**
- 1998 If present in the message to be receipted, the **/wsa:From** SOAP header block of the
- 1999 message to be receipted.
- 2000 **/osci:DeliveryReceipt/osci:ReceiptInfo/wsa:ReplyTo**
- 2001 The **/wsa:From** SOAP header block of the message to be receipted.
- 2002 **/osci:DeliveryReceipt/osci:ReceiptInfo/osci:RequestEcho ?**
- 2003 This element MUST be included, if the demand for the receipt contains the attribute
- 2004 **/osci:DeliveryReceiptDemand/@echoRequest** set to "**true**". The complete
- 2005 incoming message MUST be placed in this element in **base64Binary** format.
- 2006 To be able to proof what has been sent, the initiator in this case is strongly advised to
- 2007 encrypt the message body for himself, too.
- 2008 **/osci:DeliveryReceipt/ds:Signature**
- 2009 A digital signature of the **DeliveryReceipt** according to chapter [7.2].
- 2010 One **ds:Signature** child element **ds:Reference** MUST point to the element
- 2011 **/osci:DeliveryReceipt/ReceiptInfo** using the same-URI reference mechanism via
- 2012 the ID-attribute of this element.
- 2013 A second **/ds:Signature/ds:Reference** element MUST point to the
- 2014 **/s12:Envelope/s12:Body** block of the message to be receipted using the same-URI
- 2015 reference mechanism via the ID-attribute of the SOAP body block.
- 2016

2017 For a DeliveryReceipt, the received SOAP body block MUST to be signed "as is". The actual  
 2018 server time in UTC-format MUST be provided in  
 2019 `/osci:DeliveryReceipt/ds:Signature/ds:Object/  
 2020 xades:QualifyingProperties/xades:SignedProperties/  
 2021 xades:SignedSignatureProperties/xades/SigningTime.`

2022 If the attribute `/osci:DeliveryReceiptDemand/@qualTSPforReceipt` is set to  
 2023 "true" and can be served from this instance, the signature element MUST be extended by  
 2024 a *qualified timestamp over the signature itself*. For the timestamp itself, the specification  
 2025 [RFC3161] applies, the placement in the signature element follows **XAdES** as described in  
 2026 chapter [7.2.2]:

2027 `../ds:Signature/ds:Object/xades:QualifyingProperties/  
 2028 xades:UnsignedProperties/xades:UnsignedSignatureProperties/  
 2029 xades:SignatureTimeStamp/xades:EncapsulatedTimeStamp`

2030 If no appropriate qualified TSP-service can be provided, a fault MUST be generated to the  
 2031 requestor and the processing of the incoming message MUST be aborted.

2032 Fault 10: **QualTSPServiceNotAvailable**

2033 [Code] Sender

2034 [Subcode] QualTSPServiceNotAvailable

2035 [Reason] Requested qualified TSP service not provided by targeted node

2036 The fault [Details] property MUST outline that this timestamp was requested for a  
 2037 DeliveryReceipt and that the message is not accepted.

2038 If an incoming message of type `osci:Request` is to be receipted, the block `/osci:DeliveryReceipt`  
 2039 MUST be included as SOAP header in the corresponding `osci:Response` message.

2040 If the message to be receipted is of type `osci:Response`, the block `/osci:DeliveryReceipt` MUST  
 2041 be positioned as SOAP body of a new `osci:Request` message. This `osci:Request` message MUST be  
 2042 targeted to the endpoint denoted in `/osci:DeliveryReceiptDemand/wsa:ReplyTo`. The SOAP  
 2043 header block `/wsa:RelatesTo` of this message MUST be supplied with the `/wsa:MessageID`  
 2044 SOAP header block of the message to be receipted.

2045 **NOTE:** If a requested DeliveryReceipt cannot be produced due to processing errors or other reasons,  
 2046 an according SOAP fault MUST be generated, according to chapter [5] and the message MUST be  
 2047 discarded.

2048 **Extension made with version 2.0.1:** Additional types of receipt requests have been defined for  
 2049 message submission and message relay, see chapter [8.4.2.1]. The according receipts to be  
 2050 produced are `osci:SubmissionReceipt` and `osci:RelayReceipt`, both of the same type as  
 2051 described above for `osci:DeliveryReceipt`.

### 2052 **8.3.2.2 Reception Receipt**

2053 If requested by an `osci:ReceptionReceiptDemand` SOAP header of an `osci:Request` or  
 2054 `osci:Response` message, Reception Receipts MUST be processed after successful decryption of the  
 2055 SOAP body block. Depending on the concrete arrangement of roles in an OSCI endpoint  
 2056 implementation, it may be possible that decryption of the SOAP body and processing of a  
 2057 ReceptionReceipt demand is decoupled from the node that accepts incoming requests, respective  
 2058 responses (where DeliveryReceipt demands have to be processed immediately). Thus, a  
 2059 ReceptionReceipt is generated by Ultimate Recipient instances. For a message of type  
 2060 `osci:Response`, this is the Ultimate Recipient instance on the initiator side.

2061 The data for this type of receipt has to be placed into the following block `/osci:ReceptionReceipt`  
 2062 by the receipt generating node. The underlying schema is nearly the same as for an  
 2063 `osci:DeliveryReceipt` SOAP header block; possible attribute/element values and semantics  
 2064 differ in detail as described here.

```

2065 <osci:ReceptionReceipt @wsu:Id="xs:ID" ? >
2066 <osci:ReceiptInfo @wsu:Id="..." >
2067
2068 <wsa:MessageD>xs:anyURI </wsa:MessageD>
2069 <osci:MsgTimeStamps / >
2070 <wsa:RelatesTo / > *
2071 <osci:To> wsa:EndpointReference </osci:To>
2072 <wsa:ReplyTo> wsa:EndpointReference </wsa:ReplyTo>
2073 <wsa:From> wsa:EndpointReference </wsa:From ?
2074 <osci:RequestEcho> xs:base64Binary </RequestEcho ?
2075 </osci:ReceiptInfo>
2076 <ds:Signature / >
2077 </osci:ReceptionReceipt >

```

2078 Description of elements and attributes in the schema overview above:

2079 **/osci:ReceptionReceipt**

2080 Container holding the child elements receipt data **.../osci:ReceiptInfo** and a  
2081 **ds:Signature** element over **.../osci:ReceiptInfo**.

2082 **/osci:ReceptionReceipt/@wsu:Id**

2083 This attribute of type **xs:ID** SHOULD be provided so that unambiguous can be made to this  
2084 **/osci:ReceptionReceipt** block.

2085 **/osci:ReceptionReceipt/osci:ReceiptInfo**

2086 Container to hold the receipt details.

2087 **/osci:ReceptionReceipt/osci:ReceiptInfo/wsu:@Id**

2088 This attribute of type **xs:ID** MUST be provided; the element must be referenceable from the  
2089 signature element described below.

2090 **/osci:ReceptionReceipt/osci:ReceiptInfo/wsa:MessageID**

2091 The **/wsa:MessageID** SOAP header block of the message to be receipted.

2092 **/osci:ReceptionReceipt/osci:ReceiptInfo/osci:MsgTimeStamps**

2093 The **/osci:MsgTimeStamps** SOAP header block; this element MUST be inserted after the  
2094 receiving node that has inserted its specific timestamps according to chapter [8.1].

2095 **/osci:ReceptionReceipt/osci:ReceiptInfo/wsa:RelatesTo \***

2096 The **/wsa:RelatesTo** SOAP header blocks of the message to be receipted.

2097 **/osci:ReceptionReceipt/osci:ReceiptInfo/osci:To**

2098 This element of type **wsa:EndpointReference** denotes the destination EPR of the  
2099 message to be receipted. At least, it MUST contain the **/wsa:To** SOAP header block of this  
2100 message and those SOAP header blocks attributed by  
2101 **@wsa:IsReferenceParameter="1"**.

2102 **/osci:ReceptionReceipt/osci:ReceiptInfo/wsa:From ?**

2103 If present in the message to be receipted, this is the **/wsa:From** SOAP header block of the  
2104 message to be receipted.

2105 **/osci:ReceptionReceipt/osci:ReceiptInfo/wsa:ReplyTo**

2106 The **/wsa:ReplyTo** SOAP header block of the message to be receipted.

2107

- 2108 `/osci:ReceptionReceipt/osci:RequestEcho ?`
- 2109 This element MUST be included, if the demand for the receipt contains the attribute  
2110 `/osci:ReceptionReceiptDemand/@echoRequest` set to the value "true". The  
2111 complete incoming message MUST be placed in this element in base64Binary format.
- 2112 `/osci:ReceptionReceipt/ds:Signature`
- 2113 A digital signature of the ReceptionReceipt according to chapter [7.2.2]. The signature MUST  
2114 be generated by the Ultimate Recipient after successful decryption of the whole SOAP body  
2115 block. In case of a synchronous osci:Response to an osci:Request containing a  
2116 ReceptionReceipt demand, this is the respective UltimateRecipient node on the initiator side.  
2117 If complete decryption of the received SOAP body is not possible, a fault MUST be  
2118 generated and further message processing MUST be aborted.
- 2119 **Fault 11: MsgBodyDecryptionError**
- 2120 [Code] Sender
- 2121 [Subcode] MsgBodyDecryptionError
- 2122 [Reason] Message body decryption failed.
- 2123 The fault [Details] property MAY outline – if known to the decrypting instance - the  
2124 `ds:X509IssuerSerial` element of the certificate initially used for encryption.
- 2125 One `ds:Signature` child element `ds:Reference` MUST point to the element  
2126 `/osci:ReceptionReceipt/ReceiptInfo` using the same-URI reference mechanism via  
2127 the ID-attribute of this element.
- 2128 A second `/ds:Signature/ds:Reference` element MUST point to the  
2129 `/s12:Envelope/s12:Body` element of the message to be receipted, using the same-URI  
2130 reference mechanism via the ID-attribute of the SOAP body block. As already mentioned,  
2131 the SOAP body block in advance MUST have been successfully decrypted.
- 2132 The actual server time in UTC-format MUST be provided in the child element  
2133 `/xades:SigningTime` of `/osci:ReceptionReceipt/ds:Signature/ds:Object/  
2134 xades:QualifyingProperties/xades:SignedProperties/  
2135 xades:SignedSignatureProperties`.
- 2136 If the attribute `/osci:ReceptionReceiptDemand/@qualTSPforReceipt` is set to the  
2137 value "true" and can be served from this instance, the signature element MUST be  
2138 extended by a *qualified timestamp over the signature itself*. For the timestamp itself, the  
2139 specification [RFC3161] applies, the placement in the signature element follows **XAdES** as  
2140 described in chapter [7.2.2]:
- 2141 `.../ds:Signature/ds:Object/xades:QualifyingProperties/  
2142 xades:UnsignedProperties/  
2143 xades:UnsignedSignatureProperties/  
2144 xades:SignatureTimeStamp/xades:EncapsulatedTimeStamp`
- 2145 If no appropriate qualified TSP-service can be provided, a fault MUST be generated to the  
2146 requestor instead of the ReceptionReceipt; processing of the incoming message MAY  
2147 proceed (subject to policy to be defined for the concrete endpoint). See fault  
2148 **QualTSPServiceNotAvailable** as defined in chapter [8.3.2.1]; the fault [Details] property  
2149 MUST outline that this timestamp was requested for a ReceptionReceipt and it must state  
2150 whether further message processing takes place or not.
- 2151 The block `/osci:ReceptionReceipt` MUST be positioned as SOAP body of a new osci:Request  
2152 message. This osci:Request message MUST be targeted to the endpoint denoted in  
2153 `/osci:ReceptionReceiptDemand/wsa:ReplyTo`. The SOAP header block `/wsa:RelatesTo` of  
2154 this message MUST be supplied with the `/wsa:MessageID` SOAP header block of the message to  
2155 be receipted.

### 2156 8.3.3 Submission and Relay Receipt

2157 With version 2.0.1, additional types of receipt requests have been defined for message submission  
 2158 and message relay, see chapter [8.4.2.1]. The according receipts to be produced are  
 2159 `osci:SubmissionReceipt` and `osci:RelayReceipt`, both of the same type as described above  
 2160 for `osci:DeliveryReceipt`.

### 2161 8.3.4 Fetched Notification

2162 To request a FetchedNotification from a recipient MsgBox instance where the message is relayed, the  
 2163 following SOAP header block MUST be provided in an `osci:Request` message:

```
2164 <osci:FetchedNotificationDemand wsu:Id="..." ?  

  2165   @s12:role="http://www.osci.eu/ws/2008/05/transport/role/MsgBox" ? >  

  2166   <wsa:ReplyTo>wsa:EndpointReference</wsa:ReplyTo>  

  2167 </osci:FetchedNotificationDemand>
```

2168 Description of elements and attributes in the schema overview above:

2169 `/osci:FetchedNotificationDemand`

2170 Header block containing the demand.

2171 `/osci:FetchedNotificationDemand/@wsu:Id`

2172 For ease of referencing, this SOAP header block from WS Security SOAP header elements,  
 2173 this attribute of type `wsu:Id` SHOULD be provided.

2174 `/osci:FetchedNotificationDemand/@s12:role`

2175 This attribute of type `xs:anyURI` SHOULD<sup>23</sup> be provided with the URI outlined above. Only  
 2176 nodes acting in the role `MsgBox` are addressed by this type of demand.

2177 `/osci:ReceiptTo/wsa:ReplyTo`

2178 This required element of type `wsa:EndpointReferenceType` denotes the endpoint,  
 2179 where the requestor wishes the notification should be routed to. As `FetchedNotifications` can  
 2180 only be delivered in the SOAP body of a separate new message, this EPR SHOULD be the  
 2181 one of the `MsgBox` instance of the requestor or MAY be a specialized endpoint consuming  
 2182 notifications. The EPR MUST contain reference properties according to chapter [6,  
 2183 Addressing Endpoints]. A `.../wsa:ReferenceParameters` of the following value SHOULD  
 2184 be provided:  
 2185 `<osci:TypeOfBusinessScenario>www.osci.eu/ws/2008/05/common/urn/messageTypes/Notification/>`. In case of delivering a receipt to a `MsgBox` instance, this is  
 2186 the defaulted value for separating notification message types from other ones (see chapter  
 2187 [6, Addressing Endpoints]).  
 2188

2189 A SOAP header `/osci:FetchedNotificationDemand` MUST be processed by a node acting in  
 2190 the role of `MsgBox` when the message is pulled for the first time by the recipient of the message. It  
 2191 MUST be delivered in a separate message to the endpoint denoted in the appropriate demand. They  
 2192 MUST only be produced by `MsgBox` instances. The `/osci:FetchedNotification` block is  
 2193 positioned in the body of such a message, other body parts MUST NOT be included.

2194

<sup>23</sup> Properly, this should be a "MUST" – but has been leveraged to SHOULD for interoperability reasons. The current Microsoft WCF-implementation does not accept other URIs for `s12:role` as predefined in the SOAP 1.2 specification. This hopefully will be changed in future releases of WCF, as [SOAP12] explicitly outlines: "... other role names MAY be used as necessary to meet the needs of SOAP applications."

2195 Syntax for messages to deliver FetchedNotifications:

```

2196 <s12:Envelope ... >
2197   <s12:Header ... >
2198     ...
2199     <wsa:Action>
2200       http://www.osci.eu/2008/05/transport/urn/messageTypes/OSCIRequest
2201     </wsa:Action>
2202     <wsa:MessageID>xs:anyURI </wsa:MessageID>
2203     <wsa:RelatesTo>xs:anyURI </wsa:RelatesTo>
2204     <wsa:To>xs:anyURI </wsa:To>
2205     <osci:TypeOfBusinessScenario wsa:IsReferenceParameter="1">
2206       "http://www.osci.eu/ws/2008/05/common/urn/messageTypes/Notification"
2207     </osci:TypeOfBusinessScenario>
2208     ...
2209   </s12:Header>
2210   <s12:Body>
2211     <osci:FetchedNotification wsu:Id="..." ?>
2212     <osci:FetchedTime>xs:dateTime </osci:FetchedTime>
2213     <wsa:MessageID>xs:anyURI </wsa:MessageID>
2214     <wsa:To>wsa:Address </wsa:To>
2215     <wsa:From>wsa:EndpointReference </wsa:From>
2216     </osci:FetchedNotification>
2217   </s12:Body>
2218 </s12:Envelope>

```

2219 Description of elements and attributes in the schema overview above:

2220 **/s12:Envelope/s12:Header/wsa:Action**

2221 The value indicated herein MUST be used for that URI.

2222 **/s12:Envelope/s12:Header/wsa:MessageID**

2223 The message MUST carry a unique WS-Addressing MessageID.

2224 **/s12:Envelope/s12:Header/wsa:RelatesTo**

2225 The message MUST carry the WS-Addressing MessageID for the message a  
2226 FetchedNotification was requested for.

2227 **/s12:Envelope/s12:Header/wsa:To**

2228 The address of the destination endpoint which was stated in the EPR of the request header  
2229 element **/osci:FetchedNotificationTo/wsa:ReplyTo/wsa:Address** of the request  
2230 message.

2231 **/s12:Envelope/s12:Header/osci:TypeOfBusinessScenario**

2232 This is the instantiation of **/wsa:ReferenceParameters** bound to this EPR. It MUST be  
2233 taken from the request header element  
2234 **/osci:FetchedNotificationTo/wsa:ReplyTo/wsa:ReferenceParameters** of the  
2235 request message.

2236 **/s12:Envelope/s12:Header/osci:TypeOfBusinessScenario/**

2237 **@wsa:IsReferenceParameter**

2238 According to WS-Addressing, the element MUST be attributed with  
2239 **@wsa:IsReferenceParameter="1"**

2240 **/s12:Envelope/s12:Body/osci:FetchedNotification**

2241 Container holding the FetchedNotification.

2242

2243 `/s12:Envelope/s12:Body/osci:FetchNotification/osci:FetchTime`

2244 This element of type `xs:dateTime` MUST be set to the actual server time in UTC format  
 2245 when the `MsgBox` node processes the `FetchNotification` demand (message pulled by the  
 2246 recipient for the first time).

### 2247 **8.3.5 Additional Receipt/Notification Demand Fault Processing Rules**

2248 As fault occurrence is imaginable while processing receipt demands, it must be foreseen to  
 2249 communicate those faults to the requestor of a receipt. As far as a receipt has to be delivered directly  
 2250 in the SOAP header of a response to a request in the same HTTP-connection, such a fault occurrence  
 2251 is directly communicated to the requestor by the general SOAP/OSCI fault processing mechanisms  
 2252 and the message is discarded. **No receipt SOAP header block MUST be built up and inserted in**  
 2253 **the response in this case.**

2254 For receipts, which have to be processed in asynchronous mode and/or delivered in a separate  
 2255 `osci:Request` message, the receipt requestor has to be informed about possible fault occurrences  
 2256 asynchronously. This MUST be done by placing the fault information in the body of an `osci:Request`  
 2257 instead of the receipt block and delivered to the same endpoint the receipt message is expected at.

2258 If the message to be receipted carries a `wsa:FaultTo` SOAP header block, this is the EPR the  
 2259 `osci:Request` message carrying the fault MUST be targeted to. If this header is absent or – if present  
 2260 and carrying the value `http://www.w3.org/2005/08/addressing/anonymous` in  
 2261 `wsa:FaultTo/Address`, it MUST be targeted to the endpoint denoted in  
 2262 `/osci:ReceptionReceiptDemand/wsa:ReplyTo/Address`. The according  
 2263 `wsa:ReferenceParameters` SOAP header block MUST be

2264 `<osci:TypeOfBusinessScenario wsa:IsReferenceParameter="true">`

2265 `www.osci.eu/ws/2008/05/common/urn/messageTypes/Fault`

2266 `</osci:TypeOfBusinessScenario>`

2267 The SOAP header block `/wsa:RelatesTo` of this message MUST be supplied with the  
 2268 `/wsa:MessageID` SOAP header block of the message to be receipted.

2269 ReceptionReceipts and `FetchNotification` in general have to be delivered in asynchronous mode. If  
 2270 the request for these doesn't indicate a valid address they can be successfully targeted to, standard  
 2271 SOAP addressing error handling applies, according to [WSASOAP]; see chapter [5.2] for additional  
 2272 information.

2273 **NOTE:** Message processing MUST NOT be aborted in these situations.<sup>24</sup>

#### 2274 **8.3.5.1 Receipt Signature Validation**

2275 Receipt signatures MUST be verified by receipt consuming nodes. If signature verification fails, a fault  
 2276 MUST be generated and be made available to the source application instance initially triggering the  
 2277 corresponding receipt demand. Fault delivery in this case is a matter of implementation.

2278 Fault 12: **SignatureOfReceiptInvalid**

2279 [Code] Sender

2280 [Subcode] SignatureOfReceiptInvalid

2281 [Reason] Receipt signature verification failed.

2282 The fault [Details] property SHOULD outline the concrete verification failure. The source application  
 2283 has to handle this situation.

<sup>24</sup> Mechanisms SHOULD be considered how to inform the Initiator about the situation that requested receipts/notifications cannot be delivered. This is out of scope of this specification.

## 2284 8.4 Message Meta Data

2285 With specification version 2.0.1 a new optional SOAP header block was introduced, which is  
 2286 `oscimeta:MessageMetaData` containing information related to transport and meta data describing  
 2287 the message payload carried in the SOAP body.

2288 **NOTE:** This header block must only be handled by implementations aware for OSCI version 2.0.1 and  
 2289 higher according to the rules described in this section.

2290 This SOAP header block is intended to provide a convenient interface to the OSCI Transport  
 2291 infrastructure, carrying all meta information needed for core OSCI transport functionality itself, as well  
 2292 as those meta information needed by additional infrastructure facilities, such as logical addressing of  
 2293 end entities, message routing, selection and correlation outside the OSCI transport route, and  
 2294 application specific needs. Parts of contents of the header carry information already contained in  
 2295 headers defined by OSCI Transport Version 2.0; these are `osci:MsgTimeStamps` and those for  
 2296 demanding receipts/notifications (`osci:DeliveryReceiptDemand`,  
 2297 `osci:ReceptionReceiptDemand`, `osci:FetchedNotificationDemand`).

2298 **NOTE:** In future versions of OSCI Transport, these headers will be withdrawn<sup>25</sup>. For downwards  
 2299 compatibility, they MUST be provided and processed according to details described below. If end  
 2300 entities provide an element `oscimeta:MessageMetaData`, version 2.01 aware implementations  
 2301 MUST copy related entries to headers mentioned above and maintain timestamps semantically  
 2302 identical in both headers; processing of receipts still MUST be done based on the information  
 2303 contained in the receipts/notification demand headers.

### 2304 8.4.1 Re-used Type Definitions

2305 Prior to the description of `oscimeta:MessageMetaData` below, the following sections specify types  
 2306 used for

- 2307 • modelling logical identifiers/addresses and few, often used additional attributes for end entities
- 2308 and
- 2309 • generic modelling of diverse properties assertable to a message,

#### 2310 8.4.1.1 Non-empty String and URI

2311 These simple types are reused frequently in the scheme, restricting `xs:string` and `xs:anyURI` to  
 2312 non-empty ones:

```
2313 <xs:simpleType name="NonEmptyStringType">
2314   <xs:restriction base="xs:string">
2315     <xs:minLength value="1"/>
2316   </xs:restriction>
2317 </xs:simpleType>
```

```
2318
2319 <xs:simpleType name="NonEmptyURIType">
2320   <xs:restriction base="xs:anyURI">
2321     <xs:minLength value="1"/>
2322   </xs:restriction>
2323 /xs:simpleType>
```

#### 2324 8.4.1.2 Type Definitions used for Logical Addressing

2325 Usually, applications and user agents deal with logical identifiers for parties involved in their likewise  
 2326 processing, communication and data exchange. Resolution of these identifiers to technical addresses  
 2327 – `wsa:To` EPRs like used by OSCI – often is done by use of according service- and/or participant  
 2328 directories. Direct use of WS Addressing EPRs respective HTTP-resources has barriers with regard to  
 2329 ease of use and should be hideable on application- / user agent level.

<sup>25</sup> Proposal will be submitted for a planned open consultation process

2330 Commonly, application scenarios use different participant identifier schemes, which then are base for  
 2331 specific mapping / address resolution and routing details; a specific scheme may even determinate a  
 2332 channel/protocol to be chosen for data exchange.

2333 The `oscimeta:PartyIdentifierType` below basically models a unique party by an identifier  
 2334 value, attributed by a type outlining the scheme of this value. Optional additional attributes carry  
 2335 informational items, not used for addressing purposes.

```

2336 <xs:complexType name="PartyIdentifierType" >
2337   <xs:annotation>
2338     <xs:documentation>Value of generic party identifier, as classified by
2339     @type attribute, e.g.: Prefix:Kennung</xs:documentation>
2340   </xs:annotation>
2341   <xs:simpleContent>
2342     <xs:extension base="xs:normalizedString" >
2343       <xs:attribute name="type" type="oscimeta:NonEmptyStringType"
2344       use="required" >
2345         <xs:annotation>
2346           <xs:documentation>Orientation: ebMS Core: type, how to interpret
2347           Party-Id value, e.g.: xöV oder Justiz</xs:documentation>
2348         </xs:annotation>
2349       </xs:attribute>
2350       <xs:attribute name="name" type="oscimeta:NonEmptyStringType" >
2351         <xs:annotation>
2352           <xs:documentation>optional "friendly name" value for displaying in
2353           user agents (as e.g. known from eMail)</xs:documentation>
2354         </xs:annotation>
2355       </xs:attribute>
2356       <xs:attribute name="category" type="xs:oscimeta:NonEmptyStringType" >
2357         <xs:annotation>
2358           <xs:documentation>Concrete role of party in business scenario (e.g.
2359           "buyer", "Meldebehörde", "Standesamt"...)</xs:documentation>
2360         </xs:annotation>
2361       </xs:attribute>
2362     </xs:extension>
2363   </xs:simpleContent>
2364 </xs:complexType>
  
```

2365 Description of the type definition above:

2366 `/oscimeta:PartyIdentifierType`

2367 `xs:normalizedString` carrying the party identifier value, extended by the attributes  
 2368 below.

2369 `/oscimeta:PartyIdentifierType/@type`

2370 `oscimeta:NonEmptyStringType` outlining the scheme of this identifier value. Values to  
 2371 be agreed upon by scenarios implementing this specification based on this specific `@type`  
 2372 value.

2373 `/oscimeta:PartyIdentifierType/@name`

2374 Optional non-empty `xs:string` intended to carry a "friendly name" of the party instance for  
 2375 e.g. displaying purposes in user agents.

2376 `/oscimeta:PartyIdentifierType/@category`

2377 Optional non-empty `xs:string` intended to carry a role of the party instance in a specific  
 2378 business-/communication-scenario. Definition of values of `@category` is out of scope of  
 2379 this specification.

2380 **NOTE for implementations aware of logical addressing:**

2381 If an author respective reader for response messages does not provide WS Addressing  
 2382 header information as described in section [6], a sender respective recipient **MUST** derive  
 2383 these details via the resolution path (e.g. service- or participant-directory) bound to the  
 2384 values of the `@type` attribute above. If this value is not known by these nodes, a fault **MUST**

2385 be generated to the requestor (author or reader) and processing of the incoming message  
2386 MUST be aborted.

2387	<b>Fault 13: UnknownPartyIdentifierType</b>
2388	[Code] Sender
2389	[Subcode] Unknown PartyIdentifierType
2390	[Reason] Party identifier type not known for actual subelement <i>[name of subelement]</i> in
2391	<oscimeta:MessageMetaData>

2392 If @type attribute value is known and faults occur during resolution of the logical identifier  
2393 (e.g. directory lookup technical error or value of oscimeta:PartyIdentifierType not  
2394 found during lookup) , a fault MUST be generated to the requestor (author or reader) and  
2395 processing of the incoming message MUST be aborted.

2396	<b>Fault 14: PartyIdentifierResolutionFault</b>
2397	[Code] Sender (if identifier not found); Receiver (if technical error)
2398	[Subcode] PartyIdentifierResolutionFault
2399	[Reason] Party identifier not found or technical error when resolving actual subelement [ <i>name of subelement</i> ] in <oscimeta:MessageMetaData>
2400	
2401	[Details] Related response of service used SHOULD be given here.

2402 A further complex oscimeta:PartyType is provided, extending the type above by optional security  
2403 tokens or references to those, as well as a username/password token. This may be needed by  
2404 scenarios dealing with end entity authentication on application level or encryption/ signature to be  
2405 applied on payload level.

```

2406 <xs:complexType name="PartyType">
2407   <xs:annotation>
2408     <xs:documentation>Logical identifier and optional security tokens of
2409     that entity (binary, may carry SAML, too) </xs:documentation>
2410   </xs:annotation>
2411   <xs:sequence>
2412     <xs:element name="Identifier" type="oscimeta:PartyIdentifierType"/>
2413     <xs:element name="SecurityToken" minOccurs="0" maxOccurs="unbounded">
2414       <xs:complexType>
2415         <xs:choice>
2416           <xs:element ref="wsse:BinarySecurityToken"/>
2417           <xs:element ref="wsse:SecurityTokenReference"/>
2418           <xs:element ref="wsse:UsernameToken"/>
2419         </xs:choice>
2420       <xs:attribute name="usage">
2421         <xs:simpleType>
2422           <xs:restriction base="xs:NMTOKEN">
2423             <xs:enumeration value="AUTHENTICATION"/>
2424             <xs:enumeration value="ENCRYPTION"/>
2425             <xs:enumeration value="SIGNATURE"/>
2426           </xs:restriction>
2427         </xs:simpleType>
2428       </xs:attribute>
2429       <xs:attribute name="payloadRef" type="xs:IDREF"/>
2430     </xs:complexType>
2431   </xs:element>
2432 </xs:sequence>
2433 </xs:complexType>

```

2434 Description of the type definition above:

2435 /oscimeta:PartyType

2436       xs:sequence carrying the elements below.

2437 /oscimeta:PartyType/Identifier

- 2438 Element of type **oscimeta:PartyIdentifierType** as defined above.
- 2439 **/oscimeta:PartyType/oscimeta:SecurityToken \***
- 2440 Optional complex elements carrying security tokens used by party or references to those.
- 2441 **/oscimeta:PartyType/oscimeta:SecurityToken/@usage**
- 2442 Attribute of type **xs:NMTOKEN**, outlining the application usage of this security token.
- 2443 Enumeration values: **AUTHENTICATION | ENCRYPTION | SIGNATURE**
- 2444 **/oscimeta:PartyType/oscimeta:SecurityToken/@payloadRef**
- 2445 Optional attribute of type **xs:IDREF**. If token is carried elsewhere in the message, e.g. in the
- 2446 message body, attribute may be used to point to this element using the IDREF-mechanism.
- 2447
- 2448 **Choice for token value:**
- 2449 **/oscimeta:PartyType/oscimeta:SecurityToken/wsse:BinarySecurityToken**
- 2450 Element of type **wsse:BinarySecurityTokenType** as specified in [WSS]. If present,
- 2451 **MUST** carry a security token (e.g. SAML, X509) of the party instance.
- 2452 **/oscimeta:PartyType/oscimeta:SecurityToken/wsse:BinarySecurityToken/@ValueType**
- 2453 **ype**
- 2454 This optional attribute according to [WSS] **MUST** be provided, outlining the type of actual
- 2455 token carried here.
- 2456 **/oscimeta:PartyType/oscimeta:SecurityToken/wsse:BinarySecurityToken**
- 2457 **/@EncodingType**
- 2458 This optional attribute according to [WSS] **SHOULD** default to **#Base64Binary**. For sake of
- 2459 interoperability, this token encoding **SHOULD NOT** be changed.
- 2460 or
- 2461 **/oscimeta:PartyType/oscimeta:SecurityToken/wsse:SecurityTokenReference**
- 2462 Element provides an open content model for referencing key bearing elements as specified
- 2463 in [WSS].
- 2464 **/oscimeta:PartyType/oscimeta:SecurityToken/wsse:SecurityTokenReference**
- 2465 **/@TokenType**
- 2466 This optional attribute according to [WSS] **MUST** be provided, depicting the type of actual
- 2467 token referenced.

2468 or

2469 `/oscimeta:PartyType/oscimeta:SecurityToken/wsse:UsernameToken`

2470 Applications may require a UsernameToken as a mean of identifying the requestor by  
 2471 "username", and optionally using a password (or shared secret, or password equivalent) to  
 2472 authenticate that identity. In this case, this element according [WSS] SHOULD be provided.<sup>26</sup>

### 2473 **8.4.1.3 Property and KeyCode Type**

2474 Description of properties auf a message is based on a key/value principle, whereas the key name  
 2475 itself resides in a certain scheme. Specification of schemes, keys, values, and assigned semantics of  
 2476 properties is out of scope of this specification; code tables defined elsewhere may serve as one  
 2477 possible reference. The `oscimeta:KeyCodeType` is the model for several Meta Data elements which  
 2478 are intended to carry applicable code values in tables maintained centrally or domain-specific; these  
 2479 code tables are referenced at run-time. This type adopts artifacts of the "generic codelist" specification  
 2480 of German Administration IT-Standards, "XÖV Handbuch"<sup>27</sup>, which in turn is based on OASIS  
 2481 Genericode<sup>28</sup>. The type `xoev-dt:Code` is used with the restriction both attributes `@listURI` and  
 2482 `@listversionID` being mandatory. For details, "XÖV-Handbuch" should be considered.

```
2483 <xs:complexType name="KeyCodeType" >
2484   <xs:complexContent >
2485     <xs:restriction base="xoev-dt:Code" >
2486       <xs:sequence>
2487         <xs:element name="code" type="xs:token" form="unqualified" / >
2488         <xs:element name="name" type="xs:normalizedString"
2489           form="unqualified" minOccurs="0" / >
2490       </xs:sequence>
2491       <xs:attribute name="listURI" type="xs:anyURI" use="required" / >
2492       <xs:attribute name="listVersionID" type="xs:normalizedString"
2493         use="required" / >
2494     </xs:restriction >
2495   </xs:complexContent >
2496 </xs:complexType>
```

2497 Attribute `@listURI` MUST point to a code table instance held locally or provided at a remote location.

```
2498 <xs:complexType name="PropertyType" >
2499   <xs:sequence>
2500     <xs:element name="Key" type="oscimeta:KeyCodeType" / >
2501     <xs:element name="Value" type="oscimeta:NonEmptyStringType" / >
2502   </xs:sequence>
2503 </xs:complexType>
```

2504 Description of the type definition above:

2505 `/oscimeta:PropertyType`

2506 `xs:sequence of elements below`

2507 `/oscimeta:PropertyType/oscimeta:Key`

2508 Mandatory element of type `oscimeta:KeyCodeType`, which in turn is a sequence of  
 2509 following two elements:

2510

<sup>26</sup> Additional semantics are defined in the the OASIS specifaion Web Services Security Username Token Profile Version [WSSUSER].

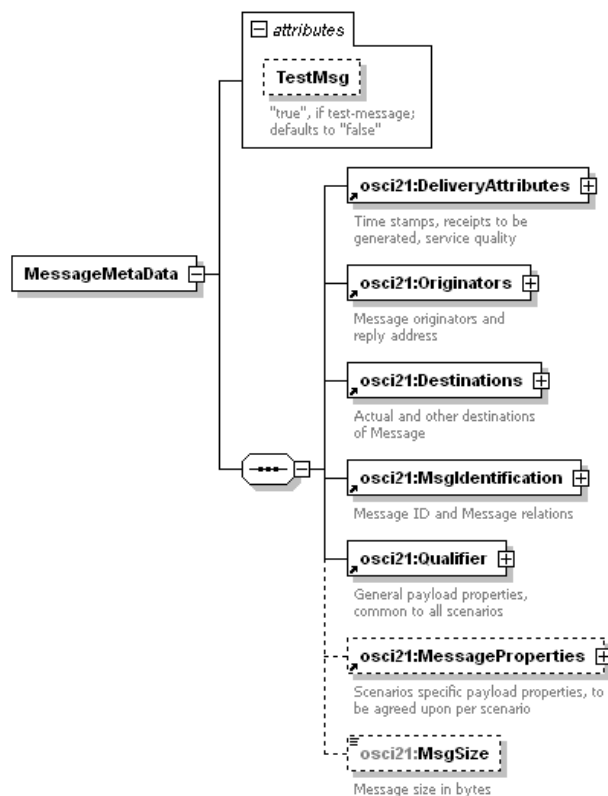
<sup>27</sup> See section 6 and 8 in [XOEVHB].

<sup>28</sup> See <http://docs.oasis-open.org/codelist/cs-genericcode-1.0/doc/oasis-code-list-representation-genericcode.html>

- 2511 `/oscimeta:PropertyType/oscimeta:Key/code`
- 2512 Mandatory element of type `xs:token` carrying the key code of this property.
- 2513 `/oscimeta:PropertyType/oscimeta:Key/name`
- 2514 Optional element of type `xs:normalizedString`; may be used as illustrative decryption of
- 2515 the key code.
- 2516 `oscimeta:KeyCodeType` has following two attributes:
- 2517 `/oscimeta:PropertyType/oscimeta:Key/@listURI`
- 2518 Mandatory attribute of type is `xs:anyURI`. Value should point to a concrete codelist instance
- 2519 or at least depict the namespace the `xoev-dt:code` value is bound to.
- 2520 `/oscimeta:PropertyType/oscimeta:Key/@listVersionID`
- 2521 Mandatory attribute of type is `xs:normalizedString`. Value should depict a concrete
- 2522 version of the codelist `@listURI` points to.
- 2523 `/oscimeta:PropertyType/oscimeta:Value`
- 2524 Mandatory element of type `oscimeta:NonEmptyStringType` carrying the key value of
- 2525 this property.
- 2526 **NOTE:** A unique key/value tuple of an element instance of type `oscimeta:PropertyType` MUST be
- 2527 given by the combination of values for `/oscimeta:Key/@listURI`, `/oscimeta:Key/code` and
- 2528 `oscimeta:Value`. The concrete version value `/oscimeta:Key/@listURI` must be outlined in
- 2529 addition.

2530 **8.4.2 Description of Message Meta Data Header**

2531 For ease of understanding, we provide a graphical overview, first.



2532

2533

Figure 9: MessageMetaData overview

2534 This header covers bricks visualized above, grouping different aspects of message meta information  
 2535 according to the annotations in Figure 9. All complex subelements carry information related to core  
 2536 transport and general payload depiction, except sub-element `oscimeta:MessageProperties`,  
 2537 which is intended to cover meta information, agreed upon in specific business scenarios, which in  
 2538 most cases would only be meaningful and processable inside these distinct scenarios.

2539 The optional Boolean attribute `@TestMsg` may be provided with the value "true" to signal a message  
 2540 exchange test scenario.

2541 To facilitate message streaming at nodes, the transport of a message is targeted to, the optional  
 2542 element `oscimeta:MsgSize` of type `xs:positiveInteger` SHOULD be set by the initially sending  
 2543 OSCI Gateway with the total message size in bytes.

#### 2544 **8.4.2.1 Delivery Attributes**

2545 This container carries transport route timestamps, required service quality and requested transport  
 2546 receipts.

```

2547 <oscimeta:DeliveryAttributes> ?
2548   <oscimeta:Origin> xs:dateTime </oscimeta:Origin> ?
2549   <oscimeta:InitialSend> xs:dateTime </oscimeta:InitialSend> ?
2550   <oscimeta:NotBefore> xs:dateTime </oscimeta:NotBefore> ?
2551   <oscimeta:ObsoleteAfter> xs:date </oscimeta:ObsoleteAfter> ?
2552   <oscimeta:Delivery> xs:dateTime </oscimeta:Delivery> ?
2553   <oscimeta:InitialFetch> xs:dateTime </oscimeta:InitialFetch> ?
2554   <oscimeta:Reception> xs:dateTime </oscimeta:Reception> ?
2555   <oscimeta:ServiceQuality> oscimeta:AnyType </oscimeta:ServiceQuality> ?
2556   <oscimeta:ReceiptRequests>
2557     <oscimeta:Submission/> ?
2558     <oscimeta:Reply/> ?
2559     <oscimeta:Delivery/> ?
2560     <oscimeta:Fetch/> ?
2561     <oscimeta:Reception/> ?
2562     <oscimeta:ReceiptTo> wsa:EndpointReference </oscimeta:ReceiptTo> ?
2563   </oscimeta:ReceiptRequests> ?
2564 </oscimeta:DeliveryAttributes>
  
```

2565 Description of elements contained in `oscimeta:DeliveryAttributes`:

2566 **NOTE:** Elements of `oscimeta:DeliveryAttributes` MUST NOT be provided or changed by  
 2567 other nodes on the message path than described here.

2568 `.../oscimeta:Origin` ?

2569 This element of type `xs:dateTime` MAY be provided by an author to denote the time a  
 2570 message is created by the author and targeted to the sender.

2571 `.../oscimeta:InitialSend`

2572 This element of type `xs:dateTime` MUST be provided by a sender to denote the time, a  
 2573 message is targeted to the receiver.

2574 `.../oscimeta:NotBefore` ?

2575 This element of type `xs:dateTime` MAY be provided by an author to denote the time a  
 2576 sender shall initiate message delivery. Senders MUST NOT initiate delivery before; if the  
 2577 value is in the past, delivery MUST be initiated immediately.

2578 `.../oscimeta:ObsoleteAfter` ?

2579 This element of type `xs:date` MAY be provided by an author to denote the date after which  
 2580 a message is to be seen as obsolete for delivery and/or consumption. If and how this  
 2581 information is handled by this endpoint this message is targeted to, is outlined in the policy of  
 2582 this endpoint; see chapter [10.2.2] for details.

- 2583 To ensure downward compatibility, the sender MUST copy the element to the corresponding  
2584 one in `osci:MsgTimeStamps`.
- 2585 `.../oscimeta:Delivery ?`
- 2586 This element of type `xs:dateTime` MUST be provided by a recipient or MsgBox node when  
2587 accepting an incoming message and MUST be set to the value of the actual time.
- 2588 To ensure downward compatibility, the recipient or MsgBox node MUST copy the element to  
2589 the corresponding one in `osci:MsgTimeStamps`.
- 2590 `.../oscimeta:InitialFetch ?`
- 2591 This element of type `xs:dateTime` MUST be provided by a MsgBox node with the value of  
2592 the actual MsgBox server time when an authorized recipient initially pulls the message from  
2593 his MsgBox instance and commits the successful initial reception of this message. This  
2594 SHOULD be done by a recipient after the first successful pulling of the message from his  
2595 MsgBox.
- 2596 This element MUST NOT be updated during subsequent pull processing on the same  
2597 message.
- 2598 To ensure downward compatibility, a MsgBox node MUST copy the element to the  
2599 corresponding one in `osci:MsgTimeStamps`.
- 2600 `.../oscimeta:Reception ?`
- 2601 This element of type `xs:dateTime` MAY be set by a recipient to his actual server time when  
2602 successfully accepting an incoming message, but it should be considered that the signature  
2603 is invalidated which was applied over SOAP header and body elements by the message  
2604 issuing instance. A recipient MAY copy the element to the corresponding one in  
2605 `osci:MsgTimeStamps`.
- 2606 It MUST be set by a MsgBox node to his actual server time when the recipient commits the  
2607 reception of a message through a `MsgBoxGetNextRequest` or `MsgBoxCloseRequest`.
- 2608 To ensure downwards compatibility, a MsgBox node MUST copy the element to the  
2609 corresponding one in `osci:MsgTimeStamps`.
- 2610 `.../oscimeta:ServiceQuality ?`
- 2611 This element of type `oscimeta:NonEmptyStringType` is intended to carry information for  
2612 a certain transport profile with requirements regarding priority, confidentiality and other  
2613 service level aspects. Specification of according properties and their semantics actually is  
2614 out of scope of this specification.<sup>29</sup>
- 2615 Details may be defined for specific application domains; according implementation  
2616 extensions must be made available in this case.
- 2617 `.../oscimeta:ReceiptRequests ?`
- 2618 Container carrying a sequence of optional empty elements to denote which types of delivery  
2619 receipts are required by the author respective sender of a message.<sup>30</sup>
- 2620 **NOTE:** In aberration to OSCI 2.0, OSCI 2.1 type of receipt requests provide no possibility to  
2621 request a qualified timestamp to be applied to the receipt as well as no demand to the

<sup>29</sup> Actually used by XTA-Project to point to “Dienstprofil” (Service quality profile) held elsewhere.

<sup>30</sup> It is to unburden an author from receipt request per individual message, it is advisable to hold standard request sets in the configuration of a sender node, may be asserted to different values of `osci21:BusinessScenario` or even `osci21:MessageType` (both elements of `osci21:MessageMetaData` explained below), to address the likewise receipt requirements. If provided different from an author, these settings MUST have precedence.

- 2622 retransmit the whole initial message in the required receipt (as defined above in section  
2623 8.3.1). These variants proved to not being used in practice and thus will be dropped finally in  
2624 a future version of this specification.
- 2625 `.../oscimeta:ReceiptRequests/oscimeta:Submission31 ?`  
2626 Empty element, to be set by author if sender shall provide a receipt upon successful  
2627 submission of the message.
- 2628 `.../oscimeta:ReceiptRequests/oscimeta:Relay ?`  
2629 Empty element, to be set by author or sender, if active nodes on the message path  
2630 forwarding the message shall provide a receipt upon successful relay.
- 2631 `.../oscimeta:ReceiptRequests/oscimeta:Delivery ?`  
2632 Empty element, to be set by author if sender indicates demand for a DeliveryReceipt. For  
2633 downward compatibility, sender MUST built up an according  
2634 `osci:DeliveryReceiptDemand` header block,  
2635 `osci:DeliveryReceiptDemand/wsa:ReplyTo` MUST be set to  
2636 `.../oscimeta:ReceiptRequest/oscimeta:ReceiptTo`, if provided; else this value  
2637 defaults to the `wsa:From` header element. Attributes defined for  
2638 `osci:DeliveryReceiptDemand` MUST NOT be set (both default to false).
- 2639 `.../oscimeta:ReceiptRequests/oscimeta:Fetch ?`  
2640 Empty element, to be set by author if sender indicates demand for a FetchedNotification. To  
2641 ensure downward compatibility, sender MUST built up an according  
2642 `osci:FetchedNotificationDemand` header block,  
2643 `osci:FetchedNotificationDemand/wsa:ReplyTo` MUST be set to  
2644 `.../oscimeta:ReceiptRequest/oscimeta:ReceiptTo`, if provided; else this value  
2645 defaults to the `wsa:From` header element.
- 2646 `.../oscimeta:ReceiptRequests/oscimeta:Reception ?`  
2647 Empty element, to be set by author if sender indicates demand for a ReceptionReceipt. For  
2648 downward compatibility, sender MUST built up an according  
2649 `osci:ReceptionReceiptDemand` header block, `osci:`  
2650 `ReceptionReceiptDemand/wsa:ReplyTo` MUST be set to  
2651 `.../oscimeta:ReceiptRequest/oscimeta:ReceiptTo`, if provided; else this value  
2652 defaults to the `wsa:From` header element. Attributes defined for  
2653 `osci:ReceptionReceiptDemand` MUST NOT be set (both default to false).
- 2654 `.../oscimeta:ReceiptRequests/oscimeta:ReceiptTo ?`  
2655 Element of type `wsa:EndpointReference`, to be set by the author. If the required Receipt  
2656 should be routed some specialized endpoint consuming receipts the EPR of this endpoint  
2657 MUST be exposed here.  
2658 If absent, the value defaults to the one for the `wsa:From` header block.

### 2659 **8.4.2.2 Originators**

2660 This complex element carries details about message author and – if different - sender.

```
2661 <osci met a: Or i gi nat or s>
2662 <osci met a: Aut hor > osci met a: Par t yType </ osci met a: Aut hor >
2663 <osci met a: Sender > osci met a: Par t yType </ osci met a: Sender > ?
2664 <osci met a: Repl yTo> osci met a: Par t yType </ osci met a: Repl yTo> ?
2665 </ osci met a: Or i gi nat or s>
```

<sup>31</sup> The format of receipts to be generated is as defined for the DeliveryReceipt as specified in chapter [8.3.2.1]. According to the `osci21:ReceiptRequest` element, the value of the receipt MUST be `osci:SubmissionReceipt` respective `osci:RelayReceipt`.

2666 Description of elements contained in `oscimeta:Originators`:

2667 **NOTE:** Elements of `oscimeta:Originators` here MUST NOT be provided or changed by other  
2668 nodes on the message path than described here.

2669 `.../oscimeta:Author ?`

2670 This element of type `oscimeta:PartyType` MUST be provided by an author of a message  
2671 respective reader when responding to it.

2672 `.../oscimeta:Sender ?`

2673 This element of type `oscimeta:PartyType` MAY be provided by a sender of a message  
2674 respective receiver when responding to it and these nodes have different ones from author  
2675 respective reader.

2676 `.../oscimeta:ReplyTo ?`

2677 This element of type `oscimeta:PartyType` MAY be provided by an author or a sender of  
2678 a message respective reader or receiver when responding to it, when replying to an address  
2679 different to the one outlined in `author` is intended.

### 2680 **8.4.2.3 Destinations**

2681 This complex element carries details about message destinations.

```
2682 <oscimeta:Destinations>
2683   <oscimeta:Reader> oscimeta:PartyType </oscimeta:Reader>
2684   <oscimeta:OtherDestinations>
2685     <oscimeta:OtherReaders> oscimeta:PartyIdentifierType
2686   </oscimeta:OtherReaders> *
2687   <oscimeta:CcReaders> oscimeta:PartyIdentifierType </oscimeta:CcReaders> *
2688   </oscimeta:OtherDestinations> ?
2689 </oscimeta:Destinations>
```

2690 Description of elements contained in `oscimeta:Destinations`:

2691 **NOTE:** Elements of `oscimeta:Destinations` here MUST NOT be provided or changed by other  
2692 nodes on the message path than Initiators.

2693 `.../oscimeta:Reader`

2694 This element of type `oscimeta:PartyType` MUST be provided by an author of a message  
2695 respective reader when responding to it. It identifies the destinations of this actual message  
2696 (or responds to such).

2697 `.../oscimeta:OtherDestinations ?`

2698 Container carrying an informational list of possible other addresses of this message.

2699 `.../oscimeta:OtherDestinations/OtherReaders *`

2700 These elements of type `oscimeta:PartyIdentifierType` MAY be provided by an  
2701 initiator of a message respective response to it, to outline other addressed readers.

2702 `.../oscimeta:OtherDestinations/CcReaders *`

2703 These elements of type `oscimeta:PartyIdentifierType` MAY be provided by an  
2704 initiator of a message respective response to it, to outline other addressed readers in  
2705 "carbon copy" role.

2706

#### 2707 8.4.2.4 *MsgIdentification*

2708 This complex element carries the application level identification of the actual message and its relation  
2709 to other ones. It refers to the following `oscimeta:ProcessIdentifierType`:

```
2710 <xs:complexType name="ProcessIdentifierType">
2711   <xs:annotation>
2712     <xs:documentation>Process ID message is related to</xs:documentation>
2713   </xs:annotation>
2714   <xs:simpleContent>
2715     <xs:extension base="oscimeta:NonEmptyStringType">
2716       <xs:attribute name="ProcessName" type="oscimeta:NonEmptyStringType">
2717         <xs:annotation>
2718           <xs:documentation>Process may have a name, e.g. "order"</xs:documentation>
2719         </xs:annotation>
2720       </xs:attribute>
2721     </xs:extension>
2722   </xs:simpleContent>
2723 </xs:complexType>
```

2725 This type is used to express the relation of a message to a distinct business process, identified by a  
2726 non-empty string.

2727 An optional attribute `@ProcessName` of type `oscimeta:NonEmptyStringType` may be used to  
2728 outline a process name.

```
2729 <oscimeta:MsgIdentification>
2730   <wsa:MessageID wsa:AttributedURIType </wsa:MessageID>
2731   <oscimeta:In-Reply-To> wsa:AttributedURIType </oscimeta:In-Reply-To> *
2732   <oscimeta:ProcessRef> ?
2733     <oscimeta:Requester> oscimeta:ProcessIdentifierType
2734   </oscimeta:Requester> ?
2735     <oscimeta:Responder> oscimeta:ProcessIdentifierType
2736   </oscimeta:Responder> ?
2737   </oscimeta:ProcessRef> ?
2738 </oscimeta:MsgIdentification>
```

2739 Description of elements contained in `oscimeta:MsgIdentification`:

2740 `.../wsa:MessageID`

2741 Mandatory element of type `wsa:AttributedURIType`. It SHOULD carry a unique  
2742 message ID (UUID) according to IETF RFC "A Universally Unique Identifier (UUID) URN  
2743 Namespace" [RFC4122]. This is the message ID as known on application level by  
2744 author/reader, which may be different from the `wsa:MessageID` used in the according WS-  
2745 Addressing header element on OSCI transport level.

2746 `.../oscimeta:In-Reply-To *`

2747 Optional elements of type `wsa:AttributedURIType`; MAY be used to identify  
2748 application-level message IDs to which the actual message is a reply.

2749 `.../oscimeta:ProcessRef ?`

2750 Optional container holding the elements described below, outlining the assignment of the  
2751 message to a certain business process ID (which may be different on service-requestor and  
2752 -responder side).

2753 `.../oscimeta:ProcessRef/Requester ?`

2754 Optional element of type `oscimeta:ProcessIdentifierType`, outlining the process  
2755 assignment on requestor side.

2756 `.../oscimeta:ProcessRef/Responder ?`

2757 Optional element of type `oscimeta:ProcessIdentifierType`, outlining the process  
2758 assignment on responder side.

### 2759 **8.4.2.5 Qualifier**

2760 This complex element carries information qualifying the message payload commonly applicable by all  
 2761 business scenarios, intended to be used as criteria for selective message processing outside the  
 2762 OSCI Transport infrastructure respective for selective message access in OSCI message boxes.

```

2763 <osci met a: Qual i f i e r >
2764   <osci met a: Subj e c t > x s : s t r i n g </osci met a: Subj e c t > ?
2765   <osci met a: Busi nessScenar i o>
2766     <osci met a: Def i ned> osci met a. KeyCodeType </osci met a: Def i ned>
2767   |
2768     <osci met a: UnDef i ned> x s : N o r m a l i z e d S t r i n g </osci met a: UnDef i ned>
2769   </osci met a: Busi nessScenar i o>
2770   <osci met a: Ser vi ce> x s : a n y U R I </osci met a: Ser vi ce>
2771   <osci met a: M e s s a g e T y p e > @osci met a: N o n E m t y U R I T y p e
2772     osci met a: Key Code Type
2773   </osci met a: M e s s a g e T y p e >
2774 </osci met a: Qual i f i e r >
  
```

2775 Description of elements contained in `oscimeta:Qualifier`:

2776 `.../oscimeta:Subject`

2777 Optional element of type `xs:string`, carrying informational text about this message (as  
 2778 known from e-mail "about")

2779 `.../oscimeta:BusinessScenario`

2780 *Choice of:*

2781 `.../oscimeta:BusinessScenario/Defined`

2782 This element of type `oscimeta:KeyCodeType` (see section [8.4.1.3]) denotes in its `code`  
 2783 subelement a known business scenario defined in an underlying defined code table, which is  
 2784 referenced at runtime according values for its attributes `/oscimeta:Defined/@listURI`  
 2785 and `/oscimeta:Defined/@listVersionID`.

2786 *or*

2787 `.../oscimeta:BusinessScenario/Undefined`

2788 For business scenarios not already defined in a code table, this alternative may be taken.  
 2789 Element is of type `xs:normalizedString`.

2790 `.../oscimeta:Service`

2791 This element of type `xs:anyURI` MUST be provided. It denotes a distinct service in a  
 2792 certain business scenario. Possible values are out of scope of this specification; they must  
 2793 be agreed upon per business scenario.

2794 `.../oscimeta:MessageType`

2795 This element is an extension of type `oscimeta:KeyCodeType` (see section [8.4.1.3])  
 2796 MUST be provided. The value of its mandatory sub-element `code` denotes the type of  
 2797 message or document carried in the payload; possible values should normally be bound to  
 2798 specific business scenarios. Code tables are out of scope of this specification; they must be  
 2799 agreed upon per business scenario.

2800 `.../oscimeta:MessageType/@payloadSchema`

2801 This mandatory attribute of type `oscimeta:NonEmptyURIType` MUST outline the XML  
 2802 schema of the actual message carried in the SOAP body.

2803

### 2804 **8.4.2.6 MessageProperties**

2805 This optional sequence of `oscimeta:Property` elements is intended to carry information qualifying  
2806 the message payload in a manner defined inside certain business scenarios.

2807 Possible names and values of properties are out of scope of this specification; they must be agreed  
2808 upon per business scenario and specified elsewhere. Existing code tables may be referred to using  
2809 URNs.

```
2810 <osci met a: Messagepr oper t i es>
2811   <osci met a: Pr oper t y>
2812     osci met a: Pr oper t yType
2813   </ osci met a: Pr oper t y> *
2814 </ osci met a: Messagepr oper t i es> ?
```

2815 Description of elements contained in `oscimeta:MessageProperties`:

2816 `.../oscimeta:Property`

2817       Optional elements of type `oscimeta:PropertyType`<sup>32</sup>, carrying a property value of a key  
2818       in a certain namespace or external codelist.

## 2819 **8.5 X.509-Token Validation on the Message Route**

2820 A custom SOAP header is defined here for carrying X.509 certificates and details per usage instance  
2821 in the referred message body block parts.

2822 Certificate validation processing SOAP nodes MUST enrich the message SOAP header block with the  
2823 gathered certificate validation results and – for processing optimization purposes – mark those usage  
2824 instances of a certificate as "checked", if validation could be processed successfully.

2825 **NOTE:** This custom SOAP header is optional. It SHOULD be provided in infrastructures able to  
2826 perform certificate validation on the message route (which means, having a node available  
2827 implementing the syntax and semantics defined her).

2828 An XML syntax to carry validation results is defined by XKMS 2/XKISS [XKMS], which is incorporated  
2829 here. The `/xkms:ValidateResult` specified in XKMS includes original validation responses from  
2830 CAs like OCSP responses and CRLs. In addition to [XKMS], extensions are defined to satisfy  
2831 requirements coming out of the German Digital Signature Act regarding certificate validation. These  
2832 extensions are optional in general, but MUST be provided from OSCI service providers in Germany.

2833 Ultimate Recipients of messages MAY rely on the validation information thus once included in the  
2834 message body. As at least the inner CA responses are verifiable, as they are carrying signatures of  
2835 respective validation responders (OCSP, CRL...). In general, it's up to each node or endpoint on the  
2836 message route to rely on the validation information found in the message or to initiate revalidation of  
2837 used certificates following own needs und trust relations.

2838 This specification enforces no rules how a node serving certificate validation obtains certificate  
2839 validation results. It SHOULD be preferred to use the services of a trusted XKMS/XKISS responder  
2840 instance, like this a `/xkms:ValidateResult` can easily be gathered by the corresponding  
2841 `/xkms:ValidateRequest`. If the used XKMS/XKISS responder is designed as a relay bridging links  
2842 to all relevant CAs concerning the overall requirements of a concrete OSCI based communication  
2843 network, the burden of administrating CA-links and serving further protocols for those links is  
2844 delegated to the XKMS/XKISS responder provider.

2845 If using an XKMS responder, it is advisable to use the advantage of compound validation request  
2846 offered by the XKMS/XKISS protocol. All validation requests for all usage instances of the certificates  
2847 exposed in the `/osci:X509TokenContainer` custom SOAP header block MAY be combined in one

<sup>32</sup> Definiton see section [8.4.1.3].

2848 compound request, which leads to a corresponding compound response. See [XKMS] for further  
2849 details.

### 2850 **8.5.1 X.509-Token Container**

2851 This chapter describes this optional custom header to carry those certificates. In addition to the token  
2852 themselves, following information is carried, which has to be provided by source- /target applications  
2853 per token-usage:

- 2854 • Where a certificate is used in the body (by IDREF); information may be useful at recipient  
2855 side when parsing a message after SOAP body block decryption and grouping together  
2856 derived body block parts with their respective certificates/validation results (at least.  
2857 validation of signatures contained in the body SHOULD happen now)
- 2858 • Application time instant (this is the time instant a certificate must be proven as valid)
- 2859 • Possibility to indicate forced online OCSP request to downstream validation service nodes  
2860 (force bypassing possible caching of once gained OCSP responses).

2861 While processing the validation, such a node supplies the following additional information:

- 2862 • A reference to the corresponding `/xkms:ValidateResult` per usage instance
- 2863 • An indicator "validated" if all usage instance of a token have successfully been validated  
2864 (note: the only indication is the fact of validation, not the result!)
- 2865 • An indicator "validation completed" when all usage instances of all carried token have  
2866 successfully been validated.

2867 As under certain circumstances it may be, that a certificate validations serving node is not able to  
2868 gather all needed `/xkms:ValidateResult(s)` completely, the latter two indicators only serve for  
2869 processing optimization – they can be used to avoid iterating through the X509 token container and  
2870 checking for outstanding `/xkms:ValidateResult(s)` by downstream nodes / endpoints on the  
2871 message route.

2872 Syntax for an optional `/osci:X509TokenContainer`:

```
2873 <osci : X509TokenCont ai ner val i dat eCompl et ed=( " t r u e" | " f a l s e" ) ? >
2874   <osci : X509TokenI n f o I d=" x s : I D"
2875     val i dat ed=( " t r u e" | " f a l s e" ) ? >
2876     <ds : X509Dat a>
2877       <ds : X509Cer t i f i cat e>
2878     </ ds : X509Dat a>
2879     <osci : TokenAppl i cat i on
2880       ocspNoCache=( " t r u e" | " f a l s e" ) ? >
2881       val i dat eResul t Ref =" x s : I DREF" ? >
2882       <osci : Ti m e l n s t ant >x s : dat eTi m e</ osci : Ti m e l n s t ant >
2883       <osci : M s g l t e m Ref >x s : I DREF</ osci : M s g l t e m Ref >
2884     </ osci : TokenAppl i cat i on> +
2885   </ osci : X509TokenI n f o > +
2886 </ osci : X509TokenCont ai ner >
```

2887 Description of elements and attributes in the schema overview above:

2888 `/osci:X509TokenContainer ?`

2889 Optional SOAP header block containing the X.509 certificates which SHOULD be validated  
2890 by a node on the message route with validation capabilities.

2891 This container SHOULD be provided by a source application together with the payload to be  
2892 placed in the message body block at OSCI gateway entry point towards applications. If  
2893 present in an incoming message, it MUST be provided to the addressed Target Application  
2894 by the recipient's OSCI gateway.

2895 `/osci:X509TokenContainer/@validateCompleted ?`

- 2896 This optional Boolean attribute MUST be provided with the value "true" by a validation  
2897 processing node, when processing was successfully passed for all application instances of  
2898 all contained items `/osci:X509TokenInfo`. It MUST NOT be provided with the value  
2899 "true" if this condition is false – i.e. only partially successful validation processing. It MUST  
2900 NOT be provided or changed by other logical instances than validation processing nodes.
- 2901 `/osci:X509TokenContainer/osci:X509TokenInfo +`
- 2902 If an `/osci:X509TokenContainer` is present, it MUST contain at least one item of this  
2903 type; content description follows here:
- 2904 `/osci:X509TokenContainer/osci:X509TokenInfo/@Id`
- 2905 This mandatory attribute of type `xs:ID` MUST be provided. As the whole  
2906 `/osci:X509TokenContainer` is initially to be generated by a source application instance,  
2907 the value must be a UUID; the UUID-value MUST NOT start with a character unlike the  
2908 `xs:ID` production rules and SHOULD therefore be preceded by a string of "uuid:". This  
2909 attribute MUST NOT be provided or changed by other logical instances than source  
2910 applications.
- 2911 `/osci:X509TokenContainer/osci:X509TokenInfo/@validated ?`
- 2912 This optional Boolean attribute of type `xs:ID` MUST be provided with a value of "true" by a  
2913 validation processing node, when processing was successfully passed for all application  
2914 instances of this item `/osci:X509TokenInfo`. It MUST NOT be provided with the value  
2915 "true" if this condition is false – i.e. only partially successful validation processing. It MUST  
2916 NOT be provided or changed by other logical instances than validation processing nodes.
- 2917 `/osci:X509TokenContainer/osci:X509TokenInfo/ds:X509Data`
- 2918 The X.509-Token of type `ds:Data` MUST be provided here by the source application  
2919 instance. Other sub-elements than `X509Certificate` foreseen in `ds:X509Data` MUST  
2920 NOT be provided.
- 2921 `/osci:X509TokenContainer/osci:X509TokenInfo/ds:X509Data/ds:X509Certificate`
- 2922 Subelement `.../ds:509Certificate` MUST be provided. It MUST NOT be provided or  
2923 changed by other logical instances than source applications.
- 2924 `/osci:X509TokenContainer/osci:X509TokenInfo/osci:TokenApplication`
- 2925 A source application MUST initially provide this container containing application details of the  
2926 X.509-Token.
- 2927 `/osci:X509TokenContainer/osci:X509TokenInfo/osci:TokenApplication/  
2928 @ocspNoCache ?`
- 2929 This optional Boolean attribute MUST be provided with a value of "true" by the source  
2930 application instance, when the downstream validation service node shall be forced  
2931 bypassing possible caching of OCSP responses while validating this certificate. If not  
2932 provided with the value "true", a validation service node MAY use caching mechanisms to  
2933 build up validation results. It MUST NOT be provided or changed by other logical instances  
2934 than a source application instance.
- 2935 `/osci:X509TokenContainer/osci:X509TokenInfo/osci:TokenApplication/  
2936 @validateResultRef ?`
- 2937 Validation processing nodes MUST provide an `xs:IDREF` here when processing was  
2938 successfully passed for this instance of `/osci:X509TokenInfo/TokenApplication`. It  
2939 must point to the related `/xkms:ValidateResult` header child element (see next  
2940 chapter). It MUST NOT be provided or changed by other logical instances than validation  
2941 processing nodes. If present, this attribute indicates, that this instance of  
2942 `/osci:X509TokenInfo/osci:TokenApplication` is validated.

2943 `/osci:X509TokenContainer/osci:X509TokenInfo/osci:TokenApplication/`  
 2944 `osci:TimeInstant`

2945 This element of type `xs:dateTime` MUST be provided by the source application instance  
 2946 and carry the token application time instant. This time instant MUST be taken as validation  
 2947 time instant by the validation processing node (see next chapter). It MUST NOT be provided  
 2948 or changed by other logical instances than source applications.

2949 `/osci:X509TokenContainer/osci:X509TokenInfo/osci:TokenApplication/`  
 2950 `osci:MsgItemRef`

2951 This element of type `xs:IDREF` MUST be provided by the source application instance and  
 2952 carry a reference to the cryptographic element in the message body where the token was  
 2953 used. It MUST NOT be provided or changed by other logical instances than source  
 2954 applications.

## 2955 8.5.2 X.509-Token Validation Results

2956 SOAP nodes, which are willing and able to process validation for X.509 certificates contained in the  
 2957 `/osci:X509TokenContainer` SOAP header block, MUST insert the processing result in SOAP  
 2958 header blocks `/xkms:CompoundResult` containing one or more `/xkms:ValidateResult`  
 2959 elements conformant to the part XKISS of the XKMS specification, see [XKMS] for details, whereby  
 2960 following profiling applies:

2961 **R1100:** Validation results MUST be signed by the generating instance. This MAY be a XKMS  
 2962 responder involved or – if no dedicated XKMS responder is used – the node generating  
 2963 the header block `/xkms:CompoundResult` containing the `/xkms:ValidateResult`  
 2964 elements. Hence, the element `/xkms:CompoundResult/ds:Signature` MUST be  
 2965 present. The subordinate signature elements `/xkms:ValidateResult/ds:Signature`  
 2966 SHOULD be omitted.

2967 **R1120:** Nodes, consuming the validation results, MUST be able to establish trust to the validation  
 2968 results generating node through the certificate used for this signature. If no trust can be  
 2969 established, these nodes MUST ignore the affected header block  
 2970 `/xkms:CompoundResult` and MUST revalidate the affected certificates using a service  
 2971 trusted by this node.

2972 **R1130:** Nodes, consuming the validation results, MUST validate the signature of the  
 2973 `/xkms:CompoundResult`. If signature validation fails, this fact MUST be logged as a  
 2974 security error including the affected header block and validation results present in this  
 2975 tThis header block MUST be ignored.

2976 For XKMS messages an abstract extension point `xkms:MessageExtension` is foreseen to carry  
 2977 additional information. German regulations as well as EU-wide efforts for alignment of interoperable  
 2978 use of electronic signatures, require detailed information on certificate quality, validity status, used  
 2979 algorithm suitability, and the validation process itself. Thus, an `/xkms:ValidateResult` SHOULD  
 2980 contain an extension block `/xkmsEU`, XML namespace `http://uri.peppol.eu/xkmsExt/v2#`  
 2981 as defined in chapter 5.3 of [XKMSEU]<sup>33</sup>.

## 2982 8.5.3 Verification of XKMS Validate Result Signatures

2983 Signatures of `xkms:CompoundResult` header elements MUST be verified by nodes consuming  
 2984 these header elements during the process of Content Data signatures. If signature verification fails, a  
 2985 fault MUST be generated and must be made available to the instance validating Content Data  
 2986 signatures. The affected `xkms:CompoundResult` header element MUST NOT be consumed,

<sup>33</sup> 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](http://www.peppol.eu). One goal is a common XKMS responder infrastructure in the EU member states. The namespace has to be seen as preliminary. The concrete XKMS extension structure is subject to further refinement in 2009.

2987 certificate validation processing MUST be reprocessed by means out of scope of this specification. It  
2988 is strongly RECOMMENDED to log this security error. Fault delivery is an implementation matter.

2989	<b>Fault 15: SignatureOfValidateResultInvalid</b>
2990	[Code] Sender
2991	[Subcode] SignatureOfValidateResultInvalid
2992	[Reason] Verification failed for XKMS validate result

2993 The fault [Details] property SHOULD outline the concrete verification failure.

## 2994 **8.6 General Processing of Custom Header Faults**

2995 Nodes a message is targeted to MUST validate the structure of the OSCI extension headers. If  
2996 syntactically invalid or not conformant to this specification, the message MUST be discarded and  
2997 following fault MUST be generated:

2998	<b>Fault 16: MsgHeaderStructureSchemaViolation</b>
2999	[Code] Sender
3000	[Subcode] MsgHeaderStructureSchemaViolation
3001	[Reason] One or more OSCI headers violate schema definitions

3002 More information SHOULD be given in the fault [Details] property, at least for the concrete header  
3003 element the error was located at in form of an XPath expression relative to the **s12: Envelope**  
3004 element.

## 3005 9 Constituents of OSCI Message Types

3006 For all OSCI message types, the SOAP header and body block assemblies are defined in this chapter.

3007 For a quick overview, constituents of each message type are illustrated in diagrams<sup>34</sup>.

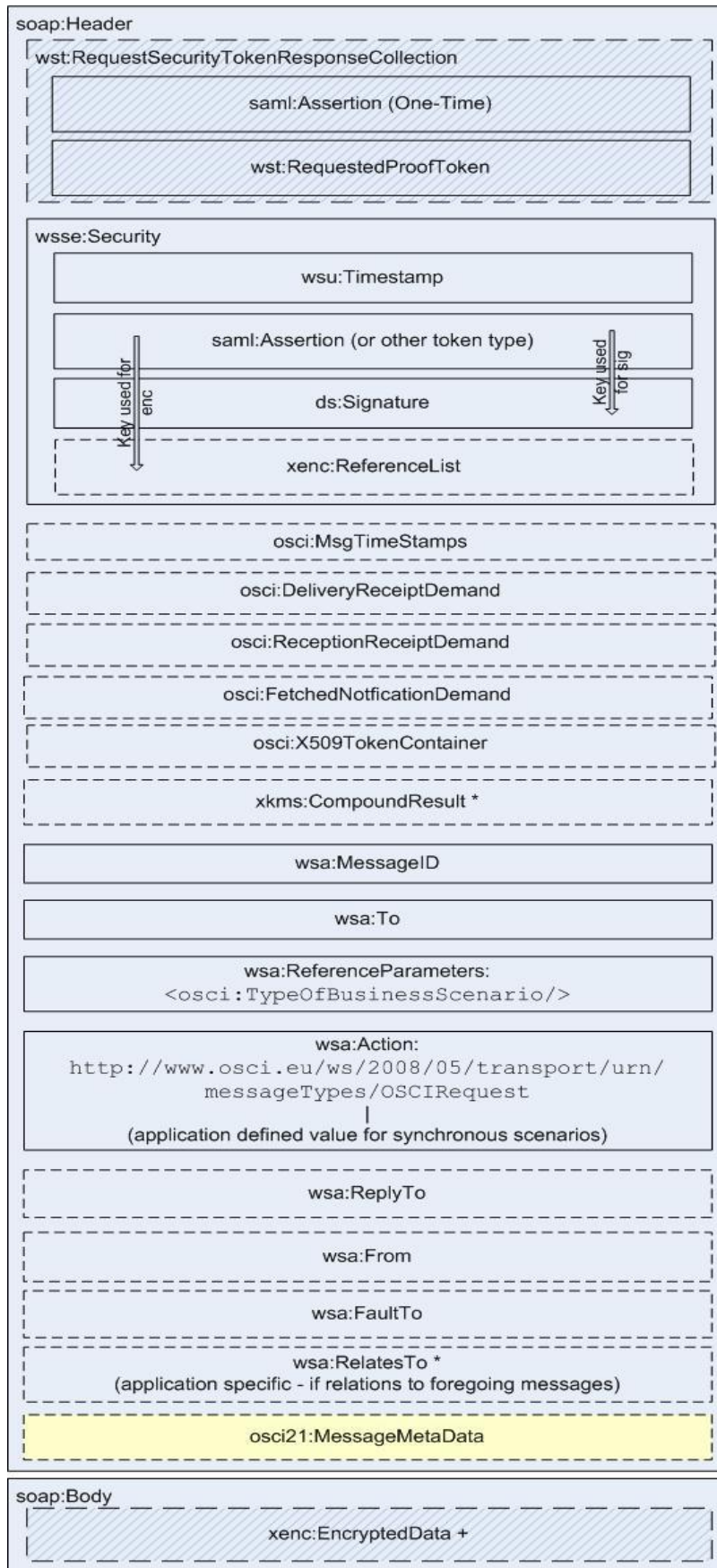
3008 **NOTE for transport security:** If applying asymmetric binding for transport security, all SOAP header  
3009 blocks as well as the SOAP body MUST be signed and encrypted. These blocks MUST be included in  
3010 the transport encryption according to chapter [7], if no symmetric binding (transport over HTTPS) is  
3011 used or the network between nodes, involved in the message transport, is secured by other  
3012 precautions.

3013

---

<sup>34</sup> All pictures have been revised in this specification version.

3014 **9.1 osci:Request**



3015

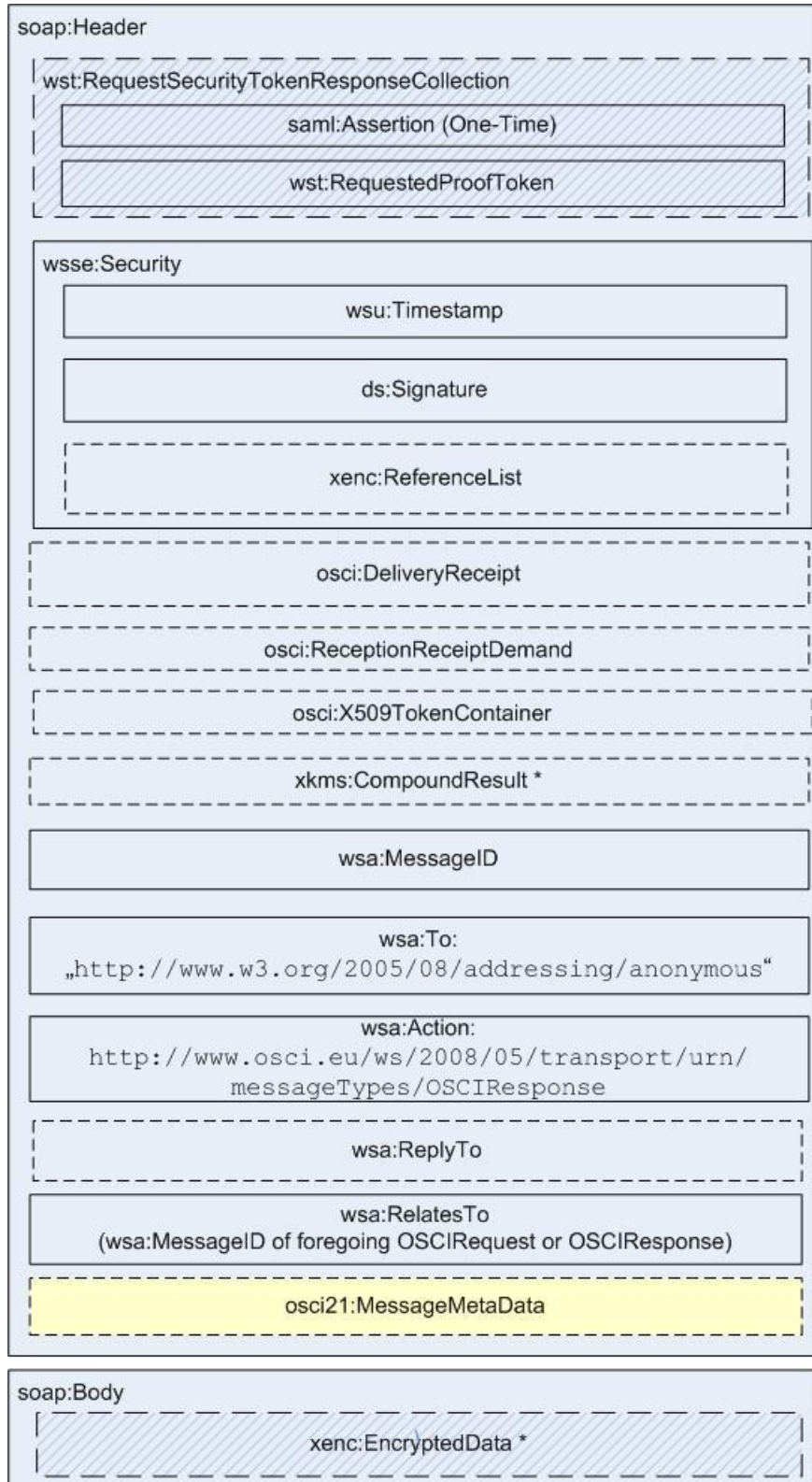
3016

Figure 10: osci:Request header and body block assembly

- 3017 SOAP header blocks:
- 3018 **/wst:RequestSecurityTokenResponseCollection ?**
- 3019 This header block carries the SAML token, which is needed for asynchronous delivery of  
3020 receipts and notification (see chapter [7.5.5] for details). It MUST only be present, if these  
3021 receipts and/or notification are required from a node in a foreign TD.
- 3022 **/wsse:Security**
- 3023 This header block MUST be present, carrying message protection data and initiator  
3024 authentication and authorization information items according to the security policy of the  
3025 node, the message is targeted to, see chapter [7.1] for details.
- 3026 **/osci:MsgTimeStamps ?**
- 3027 This optional header block MUST only be set by the initiator, if he wishes to supply a  
3028 **.../osci:ObsoleteAfter** date in here. This header block MUST be set by a MsgBox  
3029 instance and MAY be set – if not yet present - by a recipient instance. This header block  
3030 MUST always be relayed, see chapter [8.1] for details.
- 3031 **/osci:DeliveryReceiptDemand ?**
- 3032 This optional header block MUST only be set by the initiator, if he wishes to receive a  
3033 DeliveryReceipt in the backchannel response message. This header block MUST be  
3034 removed from the message by the node processing it, see chapter [8.3.1.1] for details.
- 3035 **/osci:ReceptionReceiptDemand ?**
- 3036 This optional header block MUST only be set by the initiator, if he wishes to receive a  
3037 ReceptionReceipt. This header block MUST be removed from the message by the node  
3038 processing it, see chapter [8.3.1.2] for details.
- 3039 **/osci:FetchedNotificationDemand ?**
- 3040 This optional header block MUST only be set by the initiator, if he wishes to receive a  
3041 FetchedNotification. This header block MUST only be processed by a MsgBox node instance  
3042 and MUST be removed from the message after processing it, see chapter [8.3.4] for details.
- 3043 **/osci:X509TokenContainer ?**
- 3044 This optional header block SHOULD be provided by the initiator, if he wishes to enable  
3045 certificate validation on the message route. This header block MUST always be relayed, see  
3046 chapter [8.5.1] for details.
- 3047 **/xkms:CompoundResult \***
- 3048 These optional header blocks MUST be provided by nodes processing the header block  
3049 **/osci:X509TokenContainer**. These header blocks - containing  
3050 **/xkms:ValidateResult** elements - MUST always be relayed, see chapter [8.5.2] for  
3051 details.
- 3052 **/wsa:\***
- 3053 All WS-Addressing headers MUST (if in continuous blocks) /MAY (if in dashed blocks) be  
3054 supplied by the initiator and MUST always be relayed, see chapter [6.1.2] for details.
- 3055 **/oscimeta:MessageMetaData ?**
- 3056 For version 2.0.1/2.0.2 aware implementations, this optional header block MUST be  
3057 provided by the source application, see chapter [8.4] for details. It MAY be provided  
3058 informational for version 2.0 aware implementations. This header block MUST always be  
3059 relayed.
- 3060

- 3061 SOAP body:
- 3062 Carries the request message ContentData, generally MUST be encrypted by the source
- 3063 application or initiator for the Ultimate Recipient.

## 3064 9.2 osci:Response

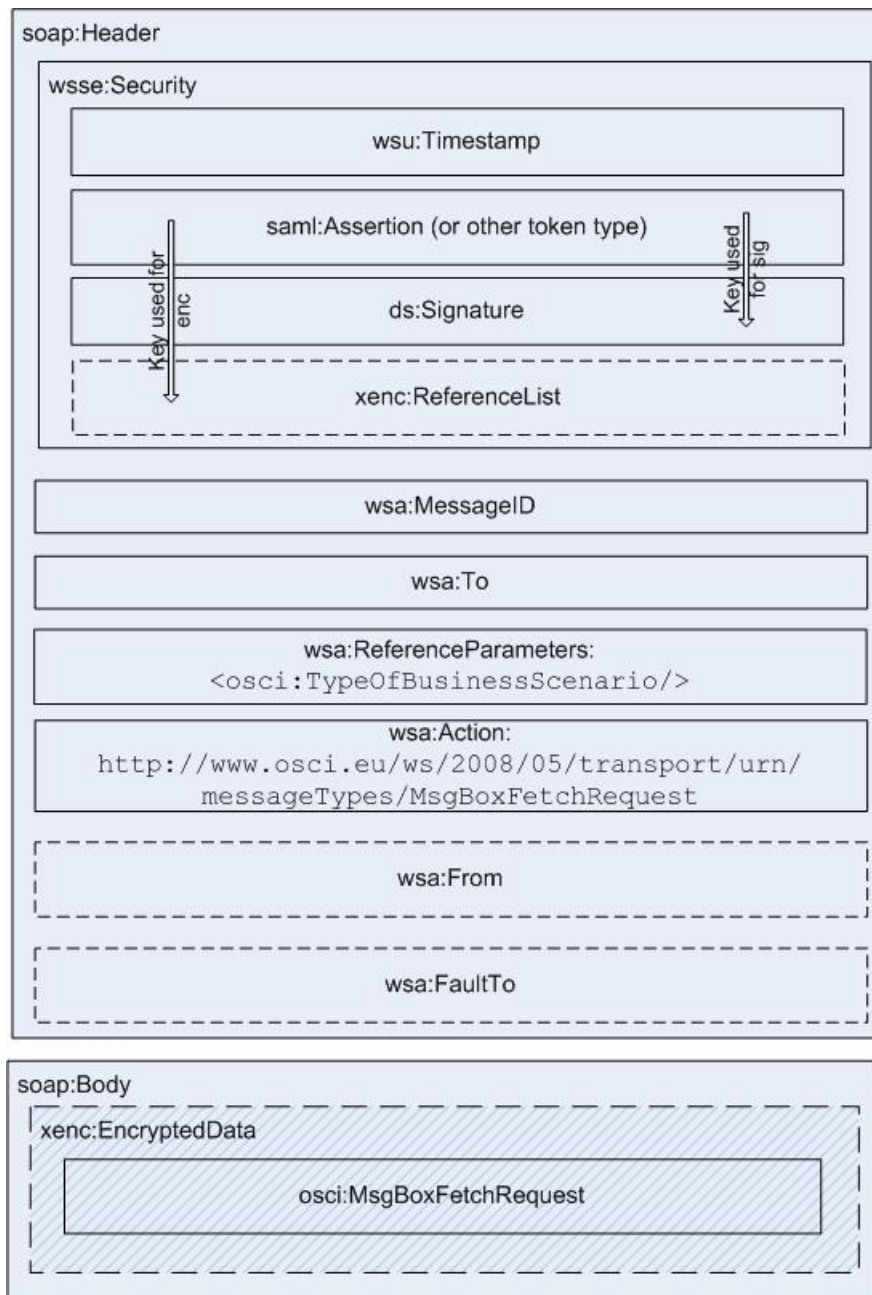


3065

3066

Figure 11: osci:Response header and body block assembly

- 3067 SOAP header blocks:
- 3068 **/wst:RequestSecurityTokenResponseCollection ?**
- 3069 This header block carries the SAML token which is needed for asynchronous delivery of  
3070 receipts and notification (see chapter [7.5.5] for details). It MUST only be present, when  
3071 these receipts and/or notification are required from a node in a foreign TD.
- 3072 **/wsse:Security**
- 3073 This header block MUST be present, carrying message protection data, see chapter [7.1] for  
3074 details.
- 3075 **/osci:DeliveryReceipt ?**
- 3076 This optional header block MUST be provided by the responding endpoint, if a demand for a  
3077 DeliveryReceipt is present in the corresponding request. This header block MUST not be  
3078 discarded or changed on the message route, see chapter [8.3.2] for details.
- 3079 **/osci:ReceptionReceiptDemand ?**
- 3080 This optional header block MUST only be set by the responding recipient, if he wishes to  
3081 receive a ReceptionReceipt for the response message. This header block MUST NOT be set  
3082 by a MsgBox instance. This header block MUST be removed from the message by the node  
3083 processing it, see chapter [8.3.1.2] for details.
- 3084 **/osci:X509TokenContainer ?**
- 3085 This optional header block SHOULD be provided by the responding recipient, if he wishes to  
3086 enable certificate validation on the message route. This header block SHOULD NOT be set  
3087 by a MsgBox instance. This header block MUST always be relayed, see chapter [8.5.1] for  
3088 details.
- 3089 **/xkms:CompoundResult \***
- 3090 These optional header blocks MUST be provided by nodes processing the header block  
3091 **/osci:X509TokenContainer**. These header blocks - containing  
3092 **/xkms:ValidateResult** elements - MUST always be relayed, see chapter [8.5.2] for  
3093 details.
- 3094 **/wsa:\***
- 3095 All WS-Addressing headers MUST (if in continuous blocks) /MAY (if in dashed blocks) be  
3096 supplied by the resending endpoint and MUST always be relayed, see chapter [6.1.2] for  
3097 details.
- 3098 **/oscimeta:MessageMetaData ?**
- 3099 For version 2.0.1 aware implementations, this optional header block MUST be provided by  
3100 the responding target application, see chapter [8.4] for details. It MAY be provided  
3101 informational for version 2.0 aware implementations. This header block MUST always be  
3102 relayed.
- 3103 SOAP body:
- 3104 May carry the response message ContentData in case of point-to-point scenarios. If present,  
3105 it generally MUST be encrypted by the target application or recipient for the initiator. If an  
3106 error occurred, a fault message is placed here instead.
- 3107

3108 **9.3 MsgBoxFetchRequest**

3109

3110

Figure 12: MsgBoxFetchRequest header and body block assembly

3111 SOAP header blocks:

3112 **/wsse:Security**

3113 This header block MUST be present, carrying message protection data and requestor  
 3114 (MsgBox owner in this case) authentication and authorization information items, according to  
 3115 the security policy MsgBox instance, see chapter [7.1] for details.

3116 **/wsa:\***

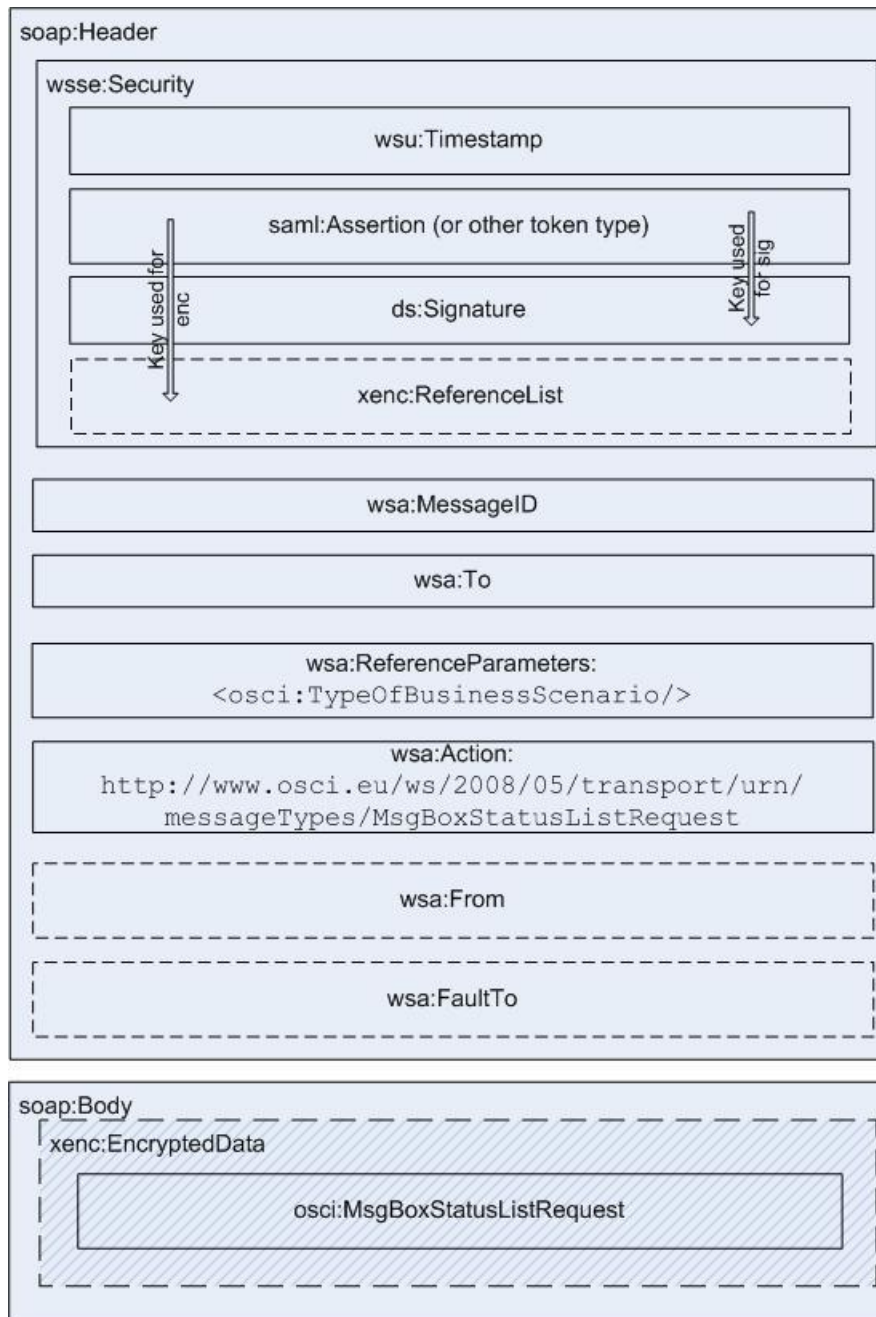
3117 All WS-Addressing headers MUST (if in continuous blocks) /MAY (if in dashed blocks) be  
 3118 supplied by the initiator, see chapter [6.1.2] and [8.2.1] for details.

3119

3120 SOAP body:

3121 Carries the details of the MsgBoxFetchRequest, generally MUST be transport encrypted,  
3122 see chapter [8.2.1] for details.

## 3123 9.4 MsgBoxStatusListRequest



3124

3125 Figure 13: MsgBoxStatusListRequest header and body block assembly

3126 SOAP header blocks:

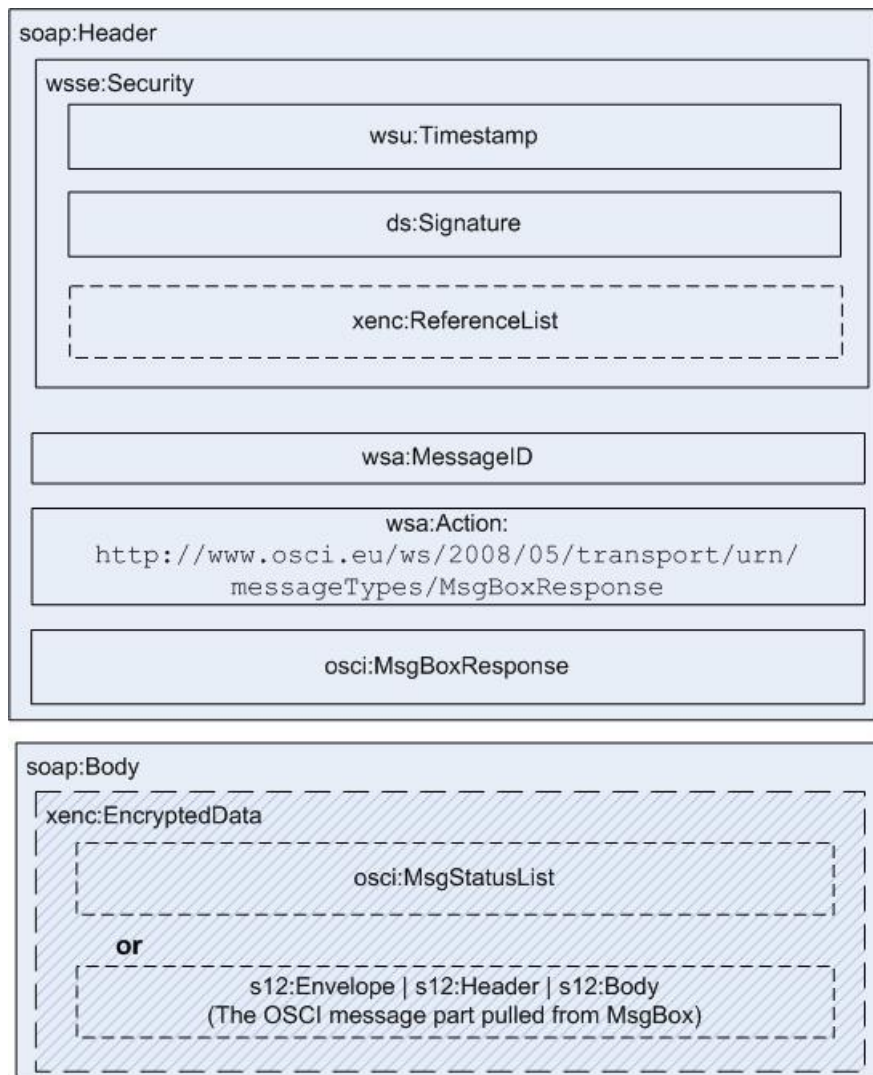
3127 **/wsse:Security**

3128 This header block MUST be present, carrying message protection data and requestor  
3129 (MsgBox owner in this case) authentication and authorization information items according  
3130 the security policy MsgBox instance, see chapter [7.1] for details.

3131

- 3132 **/wsa:\***
- 3133 All WS-Addressing headers MUST (if in continuous blocks) /MAY (if in dashed blocks) be  
3134 supplied by the initiator, see chapter [6.1.2] and [8.2.2] for details.
- 3135 SOAP body:
- 3136 Carries the details of the MsgBoxFetchRequest, generally MUST be transport encrypted,  
3137 see chapter [8.2.2] for details.

## 3138 9.5 MsgBoxResponse



3139  
3140 Figure 14: MsgBoxResponse header and body block assembly

- 3141 SOAP header blocks:
- 3142 **/wsse:Security**
- 3143 This header block MUST be present, carrying message protection data, see chapter [7.1] for  
3144 details.
- 3145 **/wsa:\***
- 3146 All WS-Addressing headers MUST (if in continuous blocks) /MAY (if in dashed blocks) be  
3147 supplied by the initiator, see chapter [6.1.2] and [8.2.3] for details.
- 3148

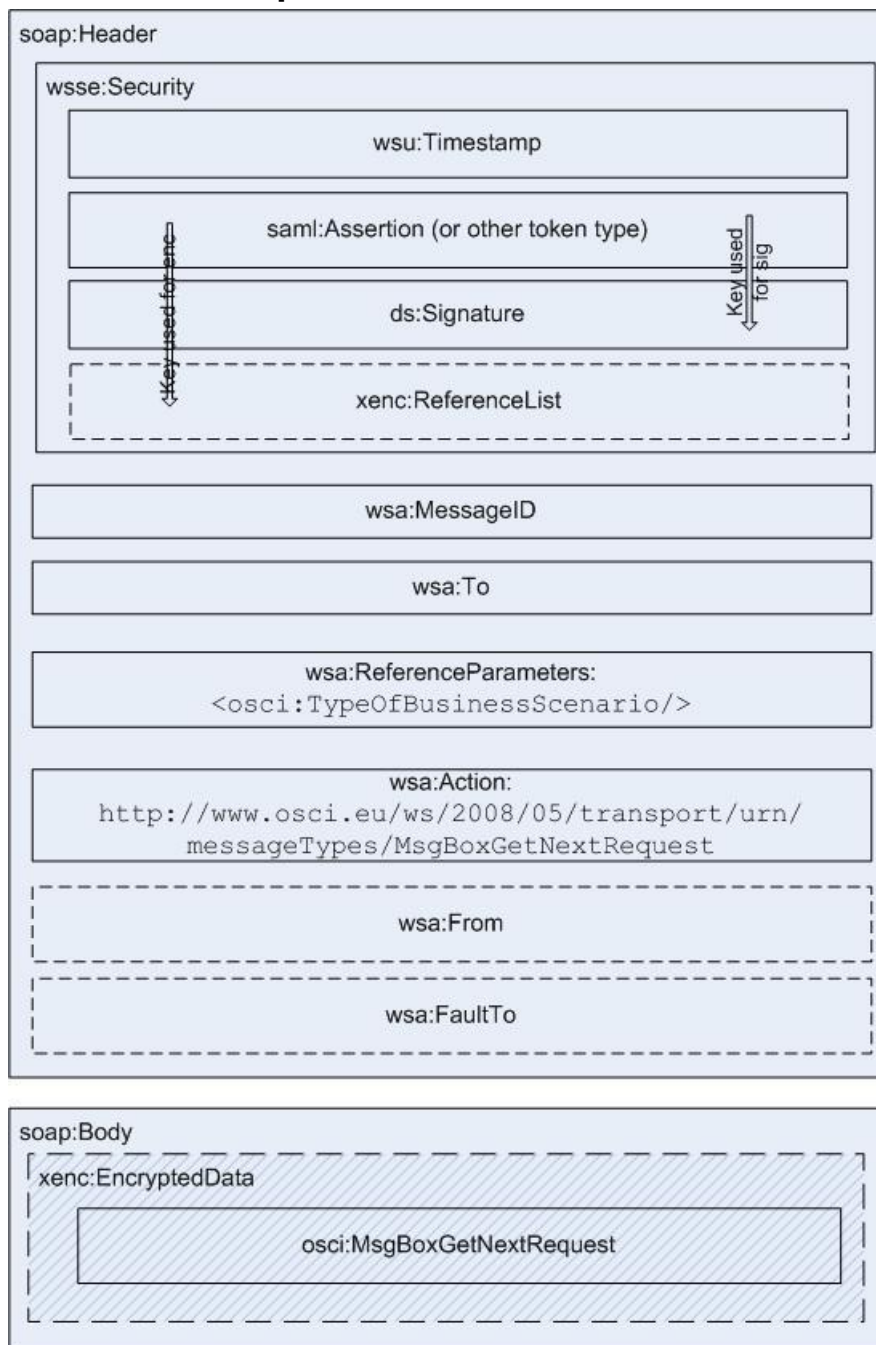
3149 **/osci:MsgBoxResponse**

3150 This header, carrying status information concerning the actual message box access, MUST  
3151 be set by the resending MsgBox instance, see chapter [8.2.3] for details.

3152 SOAP body:

3153 Carries the requested message status list or the message fetched from the MsgBox –  
3154 depending on the initial request. It generally MUST be transport encrypted, see chapter  
3155 [8.2.3.1] and [8.2.3.2] for details. If an error occurred, a fault message is placed here instead.

## 3156 9.6 MsgBoxGetNextRequest



3157

3158

3159

Figure 15: MsgBoxGetNextRequest header and body block assembly

3160 SOAP header blocks:

3161 **/wsse:Security**

3162 This header block **MUST** be present, carrying message protection data and requestor  
3163 (MsgBox owner in this case) authentication and authorization information items according  
3164 the security policy MsgBox instance, see chapter [7.1] for details.

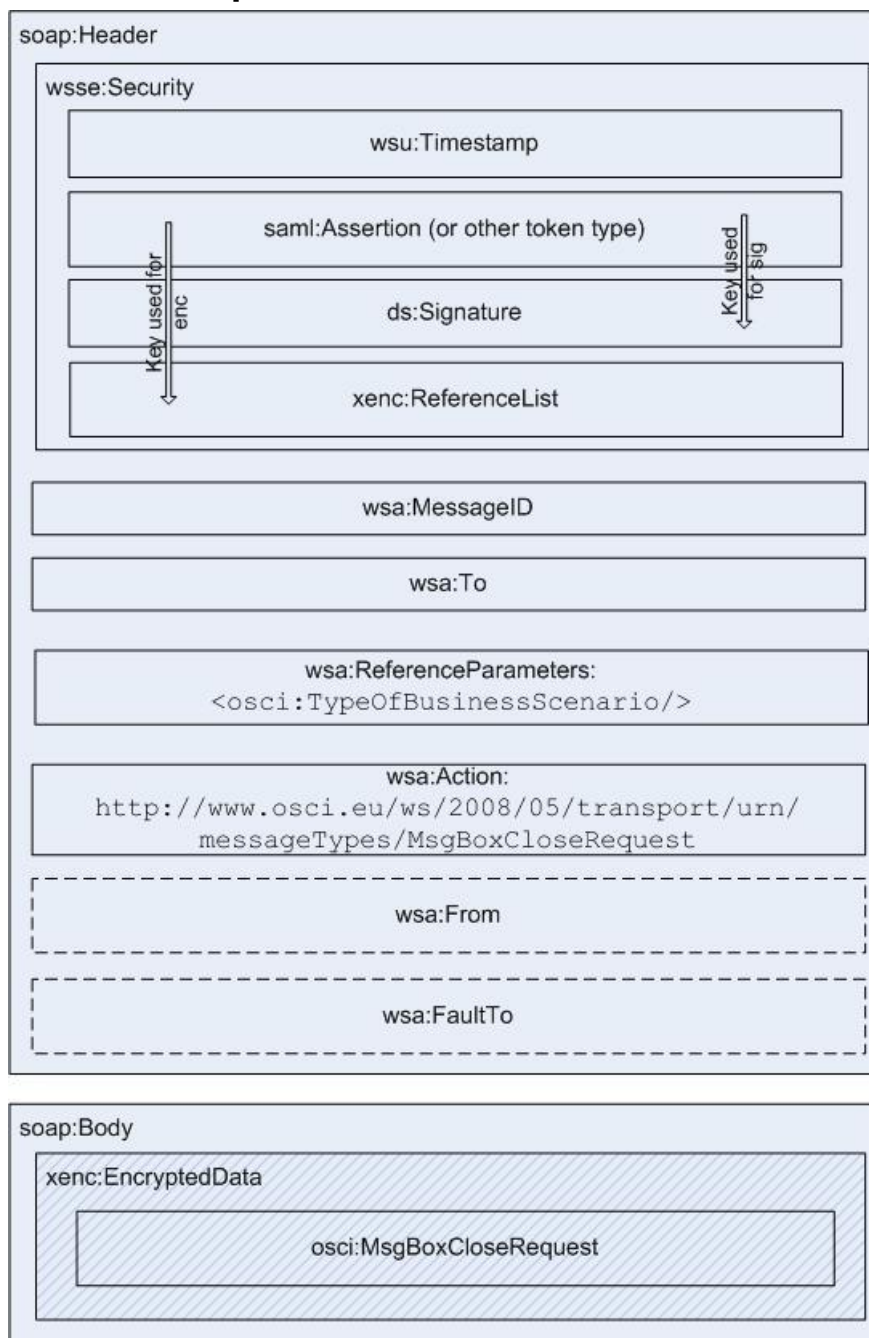
3165 **/wsa:\***

3166 All WS-Addressing headers **MUST** (if in continuous blocks) **MAY** (if in dashed blocks) be  
3167 supplied by the Initiator, see chapter [6.1.2] and [8.2.4] for details.

3168 SOAP body:

3169 Carries the details of the MsgBoxGetNextRequest, generally **MUST** be transport encrypted,  
3170 see chapter [8.2.4] for details.

3171

3172 **9.7 MsgBoxCloseRequest**3173  
3174 Figure 16: MsgBoxClose header and body block assembly

3175 SOAP header blocks:

3176 **/wsse:Security**

3177 This header block MUST be present, carrying message protection data and requestor  
 3178 (MsgBox owner in this case) authentication and authorization information items according  
 3179 the security policy MsgBox instance, see chapter [7.1] for details.

3180 **/wsa:\***

3181 All WS-Addressing headers MUST (if in continuous blocks) /MAY (if in dashed blocks) be  
 3182 supplied by the initiator, see chapter [6.1.2] and [8.2.5] for details.

3183

- 
- 3184 SOAP body:
- 3185 Carries the details of the MsgBoxCloseRequest, generally MUST be transport encrypted,
- 3186 see chapter [8.2.5] for details.

## 3187 **10 Policies and Metadata of Communication Nodes and** 3188 **Endpoints**

### 3189 **10.1 General Usage of Web Service Description Language**

3190 The Web Service Description Language (WSDL) provides a broadly adopted foundation on which  
3191 interoperable web services can be built. The WS-Policy Framework [WSPF] and WS-Policy  
3192 Attachment [WSPA] collectively define a framework, model, and grammar for expressing the  
3193 requirements and general characteristics of entities in an XML web services-based system.

3194 In general, endpoint properties and requirements MUST be described in machine readable form of  
3195 WSDLs and policies. For sake of interoperability with currently available implementations of the WS-  
3196 Stack, this specification restricts to Web Service Description Language 1.1 [WSDL11].

3197 This specification does not assume a mandatory mechanism how WSDLs of endpoints must be made  
3198 available. To facilitate retrieval and online exchange of WSDLs, they SHOULD be exposed in  
3199 appropriate directories in OSCI communication networks; the base established mechanism to access  
3200 a WSDL of a concrete web service endpoint is a HTTP(S) GET-Request in the form

3201 `http(s)://endpoint-url?WSDL.`

3202 Conformant implementations SHOULD at least support this mechanism. The specification WS  
3203 Metadata Exchange [WSMEX] describes a more sophisticated way to encapsulate services metadata  
3204 and a protocol to retrieve it. It allows the client to interact with the service automatically, fetch all  
3205 relevant metadata and aids the client in self-configuring. Support of WS Metadata Exchange is  
3206 strongly RECOMMENDED.

3207 **NOTE on WSDL/Policy integrity:** Policies and WSDL files MUST be secured by digital signatures to  
3208 allow detection of possible corruptions. Unfortunately, there is no standard format and placement  
3209 defined so far by the WS-Policy Framework for adequate digital signatures, which obviously results in  
3210 the lack of integrity check mechanisms in known framework implementations, when accessing policies  
3211 and WSDL files. An appropriate specification and recommendation for implementers will be published  
3212 by the OSCI Steering Office mid 2009 after finishing actually running tests on solution variants  
3213 addressing this issue.<sup>35</sup>

3214 Endpoints are not forced to expose their properties and requirements in form of online available and  
3215 machine readable WSDLs and/or policies. Developers may exchange this information on informal  
3216 basis out of scope of this specification (i.e. word-of-mouth, documentation).

3217 **NOTE on WSDL/Policy examples:** Patterns of WSDL instances and reference policies for classes of  
3218 OSCI based scenarios will be developed in a distinct project and be made available step by step in  
3219 2009 as addendum to this document.

3220 **Technical NOTE for policy instances:** All policies defined for an endpoint MUST carry an Id-  
3221 Attribute for the outer element `/wsp:Policy/@wsu:Id` to be referenceable for policy attachment  
3222 and metadata exchange purposes.

#### 3223 **10.1.1 WSDL and Policies for MEP Synchronous Point-To-Point**

3224 For this communication scenario, a description of endpoint requirements and abilities SHOULD be  
3225 outlined in one WSDL, containing all services and ports with their respective policies available here.  
3226 The following general requirements MUST be considered when designing WSDL instances:

3227 **R1200** - A `/wsdl111:port` MUST always contain an entry `/wsa:EndpointReference` with the  
3228 `.../wsa:Address` element as well as `.../wsa:ReferenceParameters` outlining the URI

<sup>35</sup> 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.

3229 of the `.../osci:TypeOfBusinessScenario` served by this port<sup>36</sup>. Each specific  
 3230 `/osci:TypeOfBusinessScenario` itself correlates to a concrete Content Data  
 3231 message structure given by the `/wsdl111:port` reference chain to a `/wsdl111:binding`  
 3232 and `/wsdl111:portType` entry in this WSDL instance.

## 3233 10.1.2 WSDL and Policies for Asynchronous MEPs via Message Boxes

3234 These MEPs at least have two endpoints in view, a message is targeted to:

- 3235 • Initially, a source application has to be built up the SOAP body content, according to a  
 3236 concrete schema bound to the actual underlying `/osci:TypeOfBusinessScenario`. In  
 3237 addition, security requirements bound to the recipient apply like end-to-end encryption and  
 3238 digital signatures to be applied to Content Data, which SHOULD be expressed by according  
 3239 WS Security Policy expressions.
- 3240 • For transport to the recipient's `MsgBox` instance, the WSDL and policies of this target node  
 3241 apply. For every `/osci:TypeOfBusinessScenario` accepted here, the body structure is of  
 3242 type `xenc:EncryptedData`. The WS Security Policy, if effect here, MUST NOT lead to  
 3243 initiate body description processing, as the therefore needed private encryption key is only  
 3244 known to the recipient node.

3245 As of today known WS-Framework implementations, WS Security Policies attached in the WSDL of  
 3246 the node, a message is targeted to, are completely in effect at the targeted node; it is not possible to  
 3247 bind them e.g. to a specific `s12:role` without in-depth change of processing logic of standard WS-  
 3248 Framework implementations.

3249 To solve this problem for this version of the OSCI Transport specification, the following  
 3250 recommendation applies<sup>37</sup>:

- 3251 • The `MsgBox` node exposes the WSDL and policies according to its needs on opaque body,  
 3252 transport security and authentication/authorization per accepted  
 3253 `/osci:TypeOfBusinessScenario`.
- 3254 • WSDL and policies in effect for the recipient node are referenced or contained in the  
 3255 `/wsa:EndpointReference/wsa:Metadata` element as described in chapter [6.1.1],  
 3256 bound to the `/wsdl111:port` policy attachment point.

## 3257 10.2 OSCI Specific Characteristics of Endpoints

3258 To enable OSCI endpoints to describe their requirements and capabilities, this specification defines  
 3259 OSCI policy assertions that leverage the WS-Policy Framework. In general, it is RECOMMENDED to  
 3260 attach the policy assertions defined here to a to a port [WSDL11] respective endpoint [WSDL20] policy  
 3261 subject.

### 3262 10.2.1 Certificates used for Signatures and Encryption

3263 For OSCI based message exchange, X.509v3 certificates MUST be used for the following purposes:

- 3264 • Encryption to be processed on initiator side
  - 3265 ○ End-to-end encryption of Content Data targeted from a source application to a target  
 3266 application
  - 3267 ○ Transport encryption, in cases where asymmetric encryption is required by a specific  
 3268 application scenario – in general expressed by an adequate security policy

---

<sup>36</sup> Following chapter [6.1.1], use of WS-Addressing in OSCI

<sup>37</sup> A WSDL/Policies template for this MEP as well as `MsgBox` access through the recipient will be made available as addendum immediately after publishing this specification.

- 3269 • Certificates used for signatures at recipient side (respective his MsgBox service); an initiator
- 3270 MAY – in cases of doubt - cross-check whether received signatures are generated with the
- 3271 certificates exposed in this endpoint policy (detection of possible man-in-the-middle attacks):
- 3272 ○ Signature application for OSCI receipts and possible other message parts – in cases
- 3273 where the signature must be useable for long term provableness
- 3274 ○ If offered: generation of cryptographic timestamps.

3275 Additional application purposes MAY be defined and supported by dedicated implementations.

3276 Syntax for the OSCI policy containing assertions for X.509v3 certificate usages:

```

3277 <wsp: Policy wsu: Id=" xs: ID" >
3278   <osci: X509CertificateAssertion>
3279     <wsp: ALL>
3280       <wsse: SecurityTokenReference wsu: Id=" xs: ID" ?
3281         Usage=
3282         " http://www.osci.eu/ws/2008/05/common/names/TokenUsage/e2eContentEncryption"
3283         |
3284         " http://www.osci.eu/ws/2008/05/common/names/TokenUsage/TransportEncryption"
3285         |
3286         " http://www.osci.eu/ws/2008/05/common/names/TokenUsage/ReceiptSigning"
3287         |
3288         " http://www.osci.eu/ws/2008/05/common/names/TokenUsage/TSPSigning" *
3289         osci: Role=
3290         " http://www.osci.eu/ws/2008/05/transport/role/Recipient" |
3291         " http://www.osci.eu/ws/2008/05/transport/role/MsgBox" |
3292         " http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver" * >
3293     ( <wsse: Embedded ValueType=" xs: anyURI " ? >
3294       <wsse: BinarySecurityToken wsu: Id=" xs: ID" ?
3295         ValueType=
3296         " http://docs.oasis-open.org/wss/2004/01/
3297           oasis-200401-wss-x509-token-profile-1.0#X509v3"
3298         EncodingType=
3299         " http://docs.oasis-open.org/wss/2004/01/
3300           oasis-200401-wss-soapmessage-security-1.0#Base64Binary" >
3301         xs: base64Binary
3302       </wsse: BinarySecurityToken>
3303     </wsse: Embedded> )
3304   ( <wsse: Reference URI=" xs: anyURI "
3305     ValueType=
3306     " http://docs.oasis-open.org/wss/2004/01/
3307       oasis-200401-wss-x509-token-profile-1.0#X509v3" /> )
3308   ( <wsse: KeyIdentifier wsu: Id=" xs: ID" ?
3309     ValueType=
3310     " http://docs.oasis-open.org/wss/2004/01/
3311       oasis-wss-soap-message-security-1.1#ThumbprintSHA1"
3312     EncodingType=
3313     " http://docs.oasis-open.org/wss/2004/01/
3314       oasis-200401-wss-soapmessage-security-1.0#Base64Binary" >
3315     xs: base64Binary
3316   </wsse: KeyIdentifier> )
3317   </wsse: SecurityTokenReference>
3318 </wsp: ALL>
3319 <osci: X509CertificateAssertion>
3320 </wsp: Policy>

```

3326 Description of elements and attributes in the schema overview above:

3327 **/wsp:Policy**

3328 The whole assertion MUST be embedded in a policy block according to WS Policy.

3329 **/wsp:Policy/@wsu:Id**

- 3330 To be referenceable, the policy MUST carry an attribute of type **xs:ID**.
- 3331 **/wsp:Policy/osci:X509CertificateAssertion**
- 3332 The policy block containing all assertions.
- 3333 **/wsp:Policy/osci:X509CertificateAssertion/wsp:ALL**
- 3334 Following the semantics of WS Policy Framework [WSPF], all behaviours represented by the  
3335 assertions embedded in this block are required/valid.
- 3336 **/wsp:Policy/osci:X509CertificateAssertion/wsp:ALL/wsse:SecurityTokenReferen**  
3337 **ce**
- 3338 This element defined in WS Security [WSS] MUST be used as container for a single X.509v3  
3339 certificate (or a reference to it) and its attributes.
- 3340 As all single certificate details are contained in this block, for brevity full path qualification  
3341 **/wsp:Policy/osci:X509CertificateAssertion/wsp:ALL/wsse:SecurityTokenReferen**  
3342 **ce** is symbolized by **[SingleToken]** in the following descriptions.
- 3343 **[SingleToken]/@wsu:id**
- 3344 A certificate contained/described in this policy MUST be uniquely referenceable – i.e., from  
3345 other policies describing the same endpoint. This attribute of type **xs:ID** MUST carry an  
3346 appropriate unique value.
- 3347 **[SingleToken]/@Usage**
- 3348 This attribute defines the purposes a certificate is used for, at least one of the URIs outlined  
3349 above MUST be supplied as value in this attribute of type list of **xs:anyURI**.
- 3350 Predefined usage semantics are:

Usage for	URI
End-to-end encryption of Content Data	<b>ht t p : // www. osci . eu/ ws/ 2008/ 05/ common/ names/ TokenUsage/ e2eCont ent Encr ypt i on</b>
Asymmetric transport encryption	<b>ht t p : // www. osci . eu/ ws/ 2008/ 05/ common/ names/ TokenUsage/ Tr anspor t Encr ypt i on</b>
Signature of OSCI receipts; also applicable for signatures of other message parts	<b>ht t p : // www. osci . eu/ ws/ 2008/ 05/ common/ names/ TokenUsage/ Recei pt Si gni ng</b>
Generation of cryptographic timestamps	<b>ht t p : // www. osci . eu/ ws/ 2008/ 05/ common/ names/ TokenUsage/ TSPSi gni ng</b>

3351 Table 9: OSCI X.509-Token usages

- 3352 **[SingleToken]/@osci:Role**
- 3353 This attribute defines logical roles a certificate is assigned to, for one of the URIs outlined  
3354 above a value MUST be supplied in this attribute of type list of **xs:anyURI**. Regularly, a  
3355 single certificate SHOULD NOT be assigned to more the one role; as constellations are  
3356 imaginable, where logical roles are pooled – like for a recipient and Ultimate Recipient, which  
3357 are using the same signature certificate - in these cases more than one role assignment  
3358 MAY be used.
- 3359 For example, this role attribute allows initiators to control, whether a receipt is signed with  
3360 the right certificate used by the specific receipt issuer role outlined in the receipt.

3361 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	<a href="http://www.osci.eu/ws/2008/05/transport/role/Recipient">http://www.osci.eu/ws/2008/05/transport/role/Recipient</a>
Ultimate receiver in the sense of [SOAP12]; i.e. an Ultimate Recipient using this certificate for end-to-end encryption or ReceptionReceipt signing	<a href="http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver">http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver</a>
MsgBox service node; i.e. may have his own transport encryption certificate (as MUST be used for "one time tokens")	<a href="http://www.osci.eu/ws/2008/05/common/names/role/MsgBox">http://www.osci.eu/ws/2008/05/common/names/role/MsgBox</a>

3362 Table 10: SOAP/OSCI roles assigned to token usages

3363 **R1200** - Inside a [SingleToken], the certificate itself may be embedded, referenced or identified by  
 3364 a thumbprint. Other choices foreseen by WS-Security for  
 3365 `/wsse:SecurityTokenReference` MUST NOT be used.

3366 Choice for embedded tokens

3367 `[SingleToken]/wsse:Embedded`

3368 This choice MUST be taken for embedding X.509v3 certificates. It is strongly  
 3369 RECOMMENDED to use this choice for certificates, to be used for encryption purposes, as  
 3370 an initiator and STS may need the respective public key for encryption. Referencing those  
 3371 certificates would cause additional to network connection needs.

3372 `[SingleToken]/wsse:Embedded/@ValueType`

3373 This element MAY carry this attribute of type `xs:anyURI`. It is not used in the context of this  
 3374 policy.

3375 `[SingleToken]/wsse:Embedded/wsse:BinarySecurityToken`

3376 Generic container to carry security tokens in binary format; MUST contain the X.509v3  
 3377 certificate in base64Binary format.

3378 `[SingleToken]/wsse:Embedded/wsse:BinarySecurityToken/@wsu:Id`

3379 This attribute of type `xs:ID` is optional. It is not used in the context of this policy.

3380 `[SingleToken]/wsse:Embedded/wsse:BinarySecurityToken/@ValueType`

3381 As only X.509v3 certificates are described/contained here, the URI outlined above MUST be  
 3382 supplied as value in this attribute of type of `xs:anyURI`.

3383

3384 **[SingleToken]/wsse:Embedded/wsse:BinarySecurityToken/@EncodingType**

3385 Hence X.509v3 certificates MUST be encoded in base64Binary format here, the URI outlined  
3386 above MUST be supplied as value in this attribute of type of **xs:anyURI**.

3387 Choice for directly referencing tokens

3388 **[SingleToken]/wsse:Reference**

3389 This choice MUST be taken, if referencing X.509v3 certificates stored otherwise.

3390 **[SingleToken]/wsse:Reference/@URI**

3391 This attribute of type **xs:anyURI** MUST identify an X.509v3 certificate. If a fragment is  
3392 specified, then it indicates the local ID of the security token being referenced. The URI  
3393 MUST NOT identify a **/wsse:SecurityTokenReference** element, a **/wsse:Embedded**  
3394 element, a **/wsse:Reference** element, or a **/wsse:KeyIdentifier** element.

3395 **[SingleToken]/wsse:Reference/@ValueType**

3396 As only X.509v3 certificates are described/contained here, the URI outlined above MUST be  
3397 supplied as value in this attribute of type of **xs:anyURI**.

3398 Choice for referencing tokens by thumbprint

3399 This choice SHOULD NOT be used for certificates to be used for encryption purposes, as this may  
3400 burden initiator and STS to locate the needed public key for encryption.

3401 **[SingleToken]/wsse:KeyIdentifier**

3402 **R1210:** This choice MUST be taken if referencing X.509v3 certificates by thumbprint. Other  
3403 choices foreseen by WS-Security for **/wsse:KeyIdentifier** MUST NOT be used.

3404 **[SingleToken]/wsse:KeyIdentifier/@wsu:Id**

3405 This attribute of type **xs:ID** is optional. It is not used in the context of this policy.

3406 **[SingleToken]/wsse:KeyIdentifier/@ValueType**

3407 As only thumbprints are allowed for referencing here, the URI outlined above MUST be  
3408 supplied as value in this attribute of type of **xs:anyURI**.

3409 **[SingleToken]/wsse:KeyIdentifier/@EncodingType**

3410 Hence thumbprints MUST be encoded in base64Binary format here, the URI outlined above  
3411 MUST be supplied as value in this attribute of type of **xs:anyURI**.

3412 **NOTE on usage of alternate certificates for the same purpose and role:**

3413 If more than one certificate may be used for a combination of **[SingleToken]/@osci:Role** and  
3414 **[SingleToken]/@wsse:Usage**, these policy elements **/wsse:SecurityTokenReference**  
3415 MUST be grouped in a **/wsp:ExactlyOne** container to express that only one of the alternatives may  
3416 be chosen.

## 3417 **10.2.2 Endpoint Services and Limitations**

3418 OSCI recipients respective MsgBox services MAY offer/expose the following services and limits:

- 3419 • Qualified timestamp application for signatures; this service is requestable by an initiator for  
3420 receipts;
- 3421 • Message lifetime control; this service interprets the  
3422 **/osci:MsgTimeStamps/osci:ObsoleteAfter** SOAP header element, probably set by  
3423 an initiator. This marker only makes sense in asynchronous MEPs, hence the processing  
3424 policy assigned is only of interest for MsgBox instances;
- 3425 • Maximum accepted message size and acceptance frequency per hour.

3426 Syntax for OSCI endpoint services policy assertions:

```

3427 <wsp:Policy wsu:Id="xs:ID" >
3428
3429   <osci:QualifiedTSPAssertion PolicyRef="xs:anyURI" ? > ?
3430
3431   <osci:ObsoleteAfterAssertion PolicyRef="xs:anyURI" ? > ?
3432     <osci:MsgRetainDays>
3433       xs:positiveInteger
3434     </osci:MsgRetainDays> ?
3435     <osci:WarnMsgBeforeObsolete>
3436       xs:nonNegativeInteger
3437     </osci:WarnMsgBeforeObsolete> ?
3438   </osci:ObsoleteAfterAssertion> ?
3439
3440   <osci:MsgLimitsAssertion>
3441     <osci:MaxSize>
3442       xs:positiveInteger
3443     </osci:MaxSize> ?
3444     <osci:MaxPerHour>
3445       xs:positiveInteger
3446     </osci:MaxPerHour> ?
3447   </osci:MsgLimitsAssertion> ?
3448
3449 </wsp:Policy>

```

3450 Description of elements and attributes in the schema overview above:

3451 **/wsp:Policy**

3452       The whole assertion MUST be embedded in a policy block according to WS Policy.

3453 **/wsp:Policy/@wsu:Id**

3454       To be referenceable, the policy MUST carry an attribute of type **xs:ID**.

3455 **/wsp:Policy/osci:QualifiedTSPAssertion ?**

3456       The presence of this element signals the availability of a qualified timestamp service.

3457 **/wsp:Policy/osci:QualifiedTSPAssertion/@PolicyRef ?**

3458       This optional attribute of type **xs:anyURI** SHOULD be provided and should carry a link to  
3459       i.e. human readable policies, describing terms and conditions under which this service is  
3460       made available.

3461 **/wsp:Policy/osci:ObsoleteAfterAssertion ?**

3462       The presence of this element signals the fact that this endpoint will care about a SOAP  
3463       header entry **/osci:MsgTimeStamps/osci:ObsoleteAfter**.

3464 **/wsp:Policy/osci:ObsoleteAfterAssertion/@PolicyRef ?**

3465       This optional attribute of type **xs:anyURI** MAY be provided and carry a link to i.e. human  
3466       readable policies describing terms and conditions about deletion of messages, marked to be  
3467       obsolete meanwhile.

3468 **/wsp:Policy/osci:ObsoleteAfterAssertion/MsgRetainDays ?**

3469       This optional element of type **xs:positiveInteger** SHOULD be provided to expose the  
3470       number of days, a message is still hold available after the date provided by the  
3471       .../osci:ObsoleteAfter entry.

3472

- 3473 **/wsp:Policy/osci:ObsoleteAfterAssertion/WarningBeforeObsolete ?**
- 3474 This optional element of type **xs:nonNegativeInteger** SHOULD be provided to expose  
 3475 the number of days a warning is generated before the date provided by the  
 3476 **.../osci:ObsoleteAfter** entry. Thus, an escalation procedure could be triggered for  
 3477 messages seen to be of high importance. How this warning is generated and delivered is a  
 3478 matter of implementation of this service and SHOULD be described in the terms and  
 3479 conditions policy.<sup>38</sup>
- 3480 **/wsp:Policy/osci:MsgLimitsAssertion ?**
- 3481 The presence of this element signals the fact that this endpoint has restrictions for incoming  
 3482 messages.
- 3483 **/wsp:Policy/osci:MsgLimitsAssertion/MaxSize ?**
- 3484 This optional element of type **xs:positiveInteger** outlines the maximum size in  
 3485 kilobytes for incoming messages that this endpoint accepts.
- 3486 If an incoming message exceeds this limit, it MUST be withdrawn and a fault MUST be  
 3487 returned to the targeting node:
- 3488 **Fault 17: MsgSizeLimitExceeded**
- 3489 [Code] Sender
- 3490 [Subcode] MsgSizeLimitExceeded
- 3491 [Reason] Message size exceeds policy
- 3492 **/wsp:Policy/osci:MsgLimitsAssertion/MaxPerHour ?**
- 3493 This optional element of type **xs:positiveInteger** outlines the maximum amount of  
 3494 messages accepted per hour from the same originating node<sup>39</sup>.
- 3495 If an incoming message that originates from the same targeting node, exceeds this limit, the  
 3496 message MUST be withdrawn and a fault MUST be returned to the targeting node:
- 3497 **Fault 18: MsgFrequencyLimitExceeded**
- 3498 [Code] Sender
- 3499 [Subcode] MsgFrequencyLimitExceeded
- 3500 [Reason] Message frequency per hour exceeds policy

### 3501 **10.3 WS Addressing Metadata and WS MakeConnection**

3502 Hence the use of WS-Addressing is mandatory for OSCI, an endpoint policy MUST contain WS-  
 3503 Addressing properties described here in terms of WS-Addressing Metadata [WSAM].

3504 The following policy assertions MUST be bound to the **wsdl11:port** (WSDL 2.0: endpoints) or  
 3505 **wsdl11:binding** endpoint policy subjects, which accept messages of type **osci:Request**; WS  
 3506 MakeConnection is not supported in this case:

```
3507 <wsp:Policy wsu:Id="xs:ID" ?>
3508   <wsam:Addressing>
3509     <wsp:Policy /> ?
3510   </wsam:Addressing>
3511 </wsp:Policy>
```

<sup>38</sup> This warning could e.g. be delivered in the body of an **osci:Request** to the initiator alike the **FetchNotification** message.

<sup>39</sup> No further details defined here, it is left to implementations how to define appropriate count starting and reset points

3512 This policy ascertains the use of WS-Addressing and that the endpoint requires request messages to  
 3513 use response endpoint EPRs that contain something other than the anonymous URI as the value in  
 3514 the SOAP header element `/wsa:ReplyTo/wsa:Address`.

3515

```
3516 <wsp: Policy wsu:Id="xs:ID" ?>
3517   <wsam: M CSupported / >
3518 </wsp: Policy ?
```

3519 This policy assertion MUST only be used, if WS MakeConnection is supported by this endpoint (see  
 3520 chapter [6.2]). In this case, the value of `/wsa:ReplyTo/wsa:Address` MUST be the WS-MC  
 3521 anonymous URI template.

3522 `http://docs.oasis-open.org/ws-rx/wsmc/200702/anonymous?id={unique-String}` .

3523 The following policy assertions MUST be bound to `wsdl11:ports` (WSDL 2.0: endpoints) or  
 3524 `wsdl11:binding` policy subjects, accepting messages for `MsgBox` access - these are the messages  
 3525 of type `MsgBoxFetchRequest`, `MsgBoxStatusListRequest`, `MsgBoxGetNextRequest`, and  
 3526 `MsgBoxCloseRequest`:

```
3527 <wsp: Policy wsu:Id="xs:ID" ?>
3528   <wsam Addressing>
3529     <wsp: Policy>
3530       <wsam AnonymousResponses / >
3531     <wsp: Policy>
3532   </wsam Addressing>
3533 </wsp: Policy>
```

3534 This policy ascertains the use of WS-Addressing and that the endpoint requires request messages to  
 3535 use response endpoint EPRs that carry an URI value of

3536 `"http://www.w3.org/2005/08/addressing/anonymous"`

3537 in the SOAP header element `/wsa:ReplyTo/wsa:Address`.

## 3538 10.4 WS Reliable Messaging Policy Assertions

3539 Support of WS Reliable Messaging is strongly recommended for OSCI version 2 conformant  
 3540 implementations. Adequate policy assertions MUST be used to ascertain the details of reliable  
 3541 messaging exchange. We refer to the specification [WS Reliable Messaging Policy Assertions Version](#)  
 3542 [1.1 \[WSRMP\]](#) in this point, with no further profiling.

## 3543 10.5 MTOM Policy Assertion

3544 The SOAP Message Transmission Optimization Mechanism [MTOM] MUST be supported by  
 3545 conformant implementations. The MTOM policy assertion MUST be attached to either a  
 3546 `wsdl11:binding` or `wsdl11:port` endpoint policy subject. It is expressed as

```
3547 <wsp: Policy wsu:Id="xs:ID" ?>
3548   <wsprt om Opt i m i zed M r e S e r i a l i z a t i o n / >
3549 </wsp: Policy ?
```

3550 We refer to the specification draft [MTOMP].

3551

## 3552 10.6 WS Security Profile and Policy Assertions

### 3553 10.6.1 Endpoint Policy Subject Assertions

3554 The bindings, outlined in this chapter, apply for transport level encryption and signature<sup>40</sup>.

#### 3555 10.6.1.1 Symmetric Binding

3556 The symmetric binding assertion defines details of message protection by means of WS-Security  
3557 [WSS]. In both directions from the initiator to the recipient or his MsgBox instance and backwards the  
3558 same security tokens are used for transport level encryption and signature.

3559 According to [WSSP], this assertion SHOULD apply to the endpoint policy subject `wsdl11:binding`;  
3560 it MAY apply to operation policy subject `wsdl11:binding/wsdl11:operation`.

3561 Requirements outlined in this document for message security lead to the following restrictions of  
3562 overall options defined by WS Security Policy (see [WSSP], chapter 7.4).

3563 As described in chapter [7.5, R0600], SAML Token issued by STS instances MUST be used, which  
3564 here leads to:

3565 **R1230** - This profiling restricts the usage of `wssp:ProtectionToken`; distinct  
3566 `wssp:EncryptionToken` and `wssp:SignatureToken` MUST NOT be used.

#### 3567 10.6.1.2 Asymmetric Binding

3568 For the asymmetric binding, public keys of X.509v3 certificates are used as security tokens. The  
3569 support of this binding is OPTIONAL; it MUST be used in case anonymous access is supported as  
3570 described in chapter [6.2].

3571 According to [WSSP], this assertion SHOULD apply to the endpoint policy subject `wsdl11:binding`;  
3572 it MAY apply to operation policy subject `wsdl11:binding/wsdl11:operation`.

3573 Used certificates MUST have the according key usage set; R0610 and R0620 (see chapter [7.4])  
3574 apply here and in addition:

3575 **R1240** - The node, a message is targeted to, MUST verify the validity of certificates used for  
3576 encryption; in case a value other than valid at time of usage is stated, the message MUST  
3577 be discarded and a fault MUST be generated.

3578 Fault 19: **EncryptionCertNotValid**

3579 [Code] Sender

3580 [Subcode] EncryptionCertNotValid

3581 [Reason] Encryption certificate not stated to be valid

3582 More information about the certificate validation results SHOULD be provided in the fault  
3583 [Details] property in this case. It is strongly RECOMMENDED to log such faults to be able to  
3584 detect possible security violation attacks.

3585 In the context where certificates are used by a recipient or his MsgBox node (as described in the  
3586 chapter [10.2.1]), the assertions `/wssp:RecipientEncryptionToken` and  
3587 `/wssp:RecipientSignatureToken` SHOULD point to the according certificate entries in in the  
3588 recipients metadata file.

<sup>40</sup> Note that for end-to-end encryption of content data a hybrid technique as defined in [7.3.1] must be used.

### 3589 10.6.1.3 Transport Binding

3590 The transport binding MAY be used in scenarios, in which message protection and security correlation  
3591 is provided by means other than WS-Security. We restrict to HTTPS here:

3592 **R1250** - HTTPS MUST be used, if message protection is provided by the underlying transport  
3593 protocol.

3594 This assertion MUST apply to the endpoint policy subject `wsd111:binding`.

### 3595 10.6.2 Message Policy Subject Assertions

3596 [WSSP] offers policy statements for directions, which message parts must be present and which  
3597 message parts have to be signed and encrypted. For the here presented profiling, assertions on the  
3598 SOAP header and body block level are REQUIRED, assertions on element level according to [WSSP]  
3599 MAY be used in addition.

3600 Following outlines only show the syntax of these assertions; following requirement applies:

3601 **R1260** - Concrete instances MUST enumerate the header and body blocks marked as mandatory  
3602 for presence, to be signed and/or encrypted according to definitions made per message  
3603 type in chapter [9, Constituents of OSCI Message Types].

3604 Required message parts policy assertion:

```
3605 <wsp: Pol i cy>
3606   <wsp: Exact l yOnce>
3607     <wsp: ALL>
3608       <wssp: Requi redPart s xml ns: wssp="..." ... >
3609         <wssp: Header Name=" xs: NCName" ? Namespace=" xs: anyURI " ... /> +
3610       </ wssp: Requi redPart s>
3611     </ wsp: ALL>
3612   </ wsp: Exact l yOnce>
3613 </ wsp: Pol i cy>
```

3614 Signed message parts policy assertion:

```
3615 <wsp: Pol i cy>
3616   <wsp: Exact l yOnce>
3617     <wsp: ALL>
3618       <wssp: Si gnedPart s xml ns: wssp="..." ... >
3619         <wssp: Body />
3620         <wssp: Header Name=" xs: NCName" ? Namespace=" xs: anyURI " ... /> +
3621       </ wssp: Si gnedPart s>
3622     </ wsp: ALL>
3623   </ wsp: Exact l yOnce>
3624 </ wsp: Pol i cy>
```

3625 **NOTE:** According to R1230, the SOAP body block always MUST be included in the transport  
3626 signature to ensure integrity of coherence with the message header block parts.

3627 Encrypted message parts policy assertion:

```
3628 <wsp: Pol i cy>
3629   <wsp: Exact l yOnce>
3630     <wsp: ALL>
3631       <wssp: En cr ypt edPart s xml ns: wssp="..." ... >
3632         <wssp: Body /> ?
3633         <wssp: Header Name=" xs: NCName" ? Namespace=" xs: anyURI " ... /> *
3634       </ wssp: En cr ypt edPart s>
3635     </ wsp: ALL>
3636   </ wsp: Exact l yOnce>
3637 </ wsp: Pol i cy>
```

3638 If potentially unsecured network connections are used for message exchange, the following  
3639 requirement applies:

3640 **R1270** - If the Content Data carried in the SOAP body is not encrypted end-to-end, the body block  
3641 MUST be transport encrypted.

3642 To include the required SOAP header blocks of the different OSCI message types, the following  
3643 requirement applies:

3644 **R1280** - These policy assertions MUST be bound to the message policy subject:  
 3645 wsdl11:binding/wsdl11:operation/wsdl11:input  
 3646 respective  
 3647 wsdl11:binding/wsdl11:operation/wsdl11:output

### 3648 10.6.3 Algorithm Suite Assertions

3649 In the chapters [7.2.1 and 0, restrictions are defined to suitable cryptographic algorithms, which leads  
 3650 to the following restrictions<sup>41</sup>:

3651 **R1290** - For the symmetric case, the following restriction applies for the algorithm suite assertion:

```
3652 <wsp: Pol i cy>
3653   <wssp: Al gor i t hr Sui t e>
3654     <wsp: Pol i cy>
3655       ( <wssp: Basi c256Sha256 ... /> |
3656         <wssp: Basi c192Sha256 ... /> |
3657         <wssp: Basi c128Sha256 ... /> |
3658         <wssp: Tri pl eDesSha256 ... /> )
3659     </ wsp: Pol i cy>
3660   </ wssp: Al gor i t hr Sui t e>
3661 </ wsp: Pol i cy>
```

3662 **R1300** - For the asymmetric case, the following restriction applies for the algorithm suite assertion:

```
3663 <wsp: Pol i cy>
3664   <wssp: Al gor i t hr Sui t e>
3665     <wsp: Pol i cy>
3666       ( <wssp: Basi c256Sha256Rsa15 ... /> |
3667         <wssp: Basi c192Sha256Rsa15 ... /> |
3668         <wssp: Basi c128Sha256Rsa15 ... /> |
3669         <wssp: Tri pl eDesSha256Rsa15 ... /> )
3670     </ wsp: Pol i cy>
3671   </ wssp: Al gor i t hr Sui t e>
3672 </ wsp: Pol i cy>
```

3673 The scope of these assertions is defined by its containing assertion.

3674 **R1310** - Algorithm suite assertions MUST at least be included in assertions bound to the endpoint  
 3675 policy subject `wsdl11:binding`. In addition, variations MAY be bound to subsidiary  
 3676 policy subjects, to express specific requirements.

<sup>41</sup> 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.

## 3677 **11 Applying End-to-end Encryption and Digital Signatures** 3678 **on Content Data**

3679 Predominant for OSCI is exchange of data in an authentic, confidential manner with support for legal  
3680 binding. Hence, functionalities are needed for content data end-to-end encryption and decryption,  
3681 application of digital signatures to content data, and signature validation.

3682 To ensure interoperability and conformance with the EC-Directive on Digital Signatures as well the  
3683 German Digital Signature Act and -Ordinance and underlying technical specifications, these optional  
3684 functionalities – if provided – MUST be realized conformant to the "Common PKI Specifications for  
3685 Interoperable Applications, Part 7: Signature API" [COMPKI]. This specification is a subset of the  
3686 "eCard-API Framework" [eCardAPI], based on standards worked out by the OASIS Digital Signature  
3687 Services Technical Committee [DSS].

3688 The Common PKI Signature API defines – among others – an XML interface for the following  
3689 functions:

- 3690 • SignRequest
- 3691 • VerifyRequest
- 3692 • EncryptRequest
- 3693 • DecryptRequest.

3694 API bindings are defined for C and Java; based on the XML definitions, the defined functions could  
3695 also be realized as services provided by an OSCI Gateway implementation.

3696 To use the OSCI feature of certificate validation on the message route, messages producing instances  
3697 SHOULD supply certificates used for cryptographical operations on Content Data level in a structure  
3698 described as "X.509-Token Container" in chapter [8.5.1]. This container must be carried in a message  
3699 as custom SOAP header block.

3700 On the message consuming side, the resulting custom SOAP headers `/xkms:ValidateResponse`  
3701 SHOULD be used to simplify signature verification, as the burden of connecting to CAs is delegated to  
3702 specialized nodes on the message route, see chapter [8.4] for details.

## 3703 12 Indices

### 3704 12.1 Tables

3705	Table 1: Referenced Namespaces .....	11
3706	Table 2: Predefined business scenario types.....	17
3707	Table 3: Predefined URIs for the WS Addressing Action element .....	19
3708	Table 4: Digest method: allowed algorithm identifiers .....	23
3709	Table 5: Signature method: allowed algorithm identifiers .....	23
3710	Table 6: Symmetric encryption algorithms .....	27
3711	Table 7: Security token types – support requirements .....	27
3712	Table 8: Predefined business scenario types.....	50
3713	Table 9: OSCI X.509-Token usages .....	99
3714	Table 10: SOAP/OSCI roles assigned to token usages .....	100

3715

### 3716 12.2 Pictures

3717	Figure 1: Actors and nodes involved in the message flow .....	13
3718	Figure 2: Request Security Token Message .....	31
3719	Figure 3: Request Security Token, Body for Issue Request .....	32
3720	Figure 4: Request Security Token Response Message .....	34
3721	Figure 5: Request Security Token, Body for Issue Response .....	35
3722	Figure 6: SAML 2.0 Assertion constituents .....	36
3723	Figure 7: RST for OneTimeToken .....	40
3724	Figure 8: RSTR for OneTimeToken .....	41
3725	Figure 9: MessageMetaData overview .....	72
3726	Figure 10: osci:Request header and body block assembly.....	85
3727	Figure 11: osci:Response header and body block assembly.....	87
3728	Figure 12: MsgBoxFetchRequest header and body block assembly .....	89
3729	Figure 13: MsgBoxStatusListRequest header and body block assembly .....	90
3730	Figure 14: MsgBoxResponse header and body block assembly .....	91
3731	Figure 15: MsgBoxGetNextRequest header and body block assembly.....	92
3732	Figure 16: MsgBoxClose header and body block assembly .....	94

### 3733 12.3 OSCI specific faults

3734	Fault 1: ProcessingException .....	14
3735	Fault 2: AddrWrongActionURI .....	20
3736	Fault 3: AddrWrongTypeOfBusinessScenario.....	20

3737	Fault 4: AuthnCertNotValid.....	28
3738	Fault 5: AuthnCertInvalidKeyUsage .....	28
3739	Fault 6: AuthnSecurityLevelInsufficient .....	30
3740	Fault 7: AuthnTokenFormalMismatch .....	38
3741	<b>Fault 8: MsgSelectorNotSupported</b> .....	43
3742	Fault 9: MsgBoxRequestWrongReference.....	55
3743	Fault 10: QualTSPServiceNotAvailable.....	61
3744	Fault 11: MsgBodyDecryptionError .....	63
3745	Fault 12: SignatureOfReceiptInvalid.....	66
3746	Fault 13: UnknownPartyIdentifierType .....	69
3747	Fault 14: PartyIdentifierResolutionFault .....	69
3748	Fault 15: SignatureOfValidateResultInvalid.....	83
3749	Fault 16: MsgHeaderStructureSchemaViolation .....	83
3750	Fault 17: MsgSizeLimitExceeded .....	103
3751	Fault 18: MsgFrequencyLimitExceeded .....	103
3752	Fault 19: EncryptionCertNotValid .....	105
3753		
3754	<b>12.4 Listings</b>	
3755	Listing 1: ExampleEndpointOSCIPolicy.xml.....	129
3756		

## 3757 13 References

### 3758 13.1 Normative

- 3759 [AlgCat] Bekanntmachung zur elektronischen Signatur nach dem Signaturgesetz und der  
3760 Signaturverordnung (Übersicht über geeignete Algorithmen), Bundesnetzagentur für  
3761 Elektrizität, Gas, Telekommunikation, Post und Eisenbahnen, 17. November 2008,  
3762 <http://www.bundesnetzagentur.de/media/archive/14953.pdf>
- 3763 [COMPKI] Common PKI Specifications for interoperable Applications, Version 2.0, 20 January  
3764 2009; [http://www.common-pki.org/uploads/media/Common-](http://www.common-pki.org/uploads/media/Common-PKI_v2.0.pdf)  
3765 [PKI\\_v2.0.pdf](http://www.common-pki.org/uploads/media/Common-PKI_v2.0.pdf)
- 3766 [DSS] Digital Signature Service Core - Protocols, Elements, and Bindings Version 1.0,  
3767 OASIS Standard, 11 April 2007; [http://www.oasis-](http://www.oasis-open.org/specs/index.php#dssv1.0)  
3768 [open.org/specs/index.php#dssv1.0](http://www.oasis-open.org/specs/index.php#dssv1.0)
- 3769 [eCardAPI] Das eCard-API-Framework (BSI TR-03112). Version 1.0, Federal Office for  
3770 Information Security (Bundesamt für Sicherheit in der Informationstechnik), March  
3771 2008, <http://www.bsi.de/literat/tr/tr03112/index.htm>
- 3772 [MTOM] SOAP Message Transmission Optimization Mechanism, W3C Recommendation  
3773 25 January 2005, <http://www.w3.org/TR/soap12-mtom/>
- 3774 [MTOMP] MTOM Policy Assertion 1.1, W3C Working Draft 18 September 2007,  
3775 <http://www.w3.org/TR/soap12-mtom-policy/>
- 3776 [PKCS#1] B. Kaliski, J. Staddon: PKCS #1: RSA Cryptography Specifications – Version 2.0,  
3777 IETF RFC 2437, The Internet Society October 1998,  
3778 <http://www.ietf.org/rfc/rfc2437.txt>
- 3779 [RFC2119] S. Bradner, Key words for use in RFCs to Indicate Requirement Levels , RFC 2119,  
3780 Harvard University, March 1997, <http://www.ietf.org/rfc/rfc2119.txt>
- 3781 [RFC2396] T. Berners-Lee, R. Fielding, U.C. Irvine, L. Masiner, Uniform Resource Identifiers  
3782 (URI): Generic Syntax, RFC 2396, The Internet Society 1998;  
3783 <http://www.ietf.org/rfc/rfc2396.txt>
- 3784 [RFC3161] D. Pinkas, R. Zuccherato, Time-Stamp Protocol (TSP), IETF RFC 1661,  
3785 <http://www.ietf.org/rfc/rfc3161.txt>
- 3786 [RFC4122] A Universally Unique Identifier (UUID) URN Namespace, The Internet Engineering  
3787 Task Force July 2005, <http://www.ietf.org/rfc/rfc4122.txt>
- 3788 [SAFE] S.A.F.E. (Secure Access to Federated e-Justice/e-Government) / D.I.M. (Distributed  
3789 Identity Management), Unterarbeitsgruppe SAFE der BLK Arbeitsgruppe ITStandards  
3790 in der Justiz, April 2008, [http://www.justiz.de/ERV/Grob-](http://www.justiz.de/ERV/Grob-_und_Feinkonzept/index.php)  
3791 [\\_und\\_Feinkonzept/index.php](http://www.justiz.de/ERV/Grob-_und_Feinkonzept/index.php)
- 3792 [SAMLAC] Authentication Context for the OASIS Security Assertion Markup Language (SAML)  
3793 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)  
3794 [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)
- 3795 [SAML1] Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML)  
3796 V1.2, OASIS Standard, 2 September 2003, [http://www.oasis-](http://www.oasis-open.org/committees/download.php/3406/oasis-sstc-saml-core-1.1.pdf)  
3797 [open.org/committees/download.php/3406/oasis-sstc-saml-core-](http://www.oasis-open.org/committees/download.php/3406/oasis-sstc-saml-core-1.1.pdf)  
3798 [1.1.pdf](http://www.oasis-open.org/committees/download.php/3406/oasis-sstc-saml-core-1.1.pdf)
- 3799 [SAML2] Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML)  
3800 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)  
3801 [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)

- 3802 [SOAP12] SOAP Version 1.2 Part 1: Messaging Framework (Second Edition), W3C  
3803 Recommendation 27 April 2007, <http://www.w3.org/TR/soap12-part1/>
- 3804 [TR02102-2] Kryptographische Verfahren: Empfehlungen und Schlüssellängen; Teil 2 –  
3805 Verwendung von Transport Layer Security (TLS), Bundesamt für Sicherheit in der  
3806 Informationstechnik (BSI) Technische Richtlinie TR-02102-2, 12.2.2014,  
3807 [https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen](https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR02102/BSI-TR-02102-2_pdf.pdf)  
3808 [en/TechnischeRichtlinien/TR02102/BSI-TR-02102-2\\_pdf.pdf](https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR02102/BSI-TR-02102-2_pdf.pdf)
- 3809 [TR03107-1] Technical Guideline TR-03107-1 Electronic Identities and Trust Services in E-  
3810 Government; Part 1: Assurance levels and mechanisms, Bundesamt für Sicherheit in  
3811 der Informationstechnik (BSI) 9.4.2014, English and German version can be found at:  
3812 [https://www.bsi.bund.de/DE/Publikationen/TechnischeRichtlinien/](https://www.bsi.bund.de/DE/Publikationen/TechnischeRichtlinien/tr03107/index_htm.html)  
3813 [tr03107/index\\_htm.html](https://www.bsi.bund.de/DE/Publikationen/TechnischeRichtlinien/tr03107/index_htm.html)
- 3814 [TR03130-1] Technical Guideline TR-03130-1 Technical Guideline eID-Server; Part 1: Functional  
3815 Specification, Bundesamt für Sicherheit in der Informationstechnik (BSI) 15.1.2014,  
3816 [https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen](https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR03130/TR-03130_TR-eID-Server_Part1_pdf.pdf)  
3817 [en/TechnischeRichtlinien/TR03130/TR-03130\\_TR-eID-](https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR03130/TR-03130_TR-eID-Server_Part1_pdf.pdf)  
3818 [Server\\_Part1\\_pdf.pdf](https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR03130/TR-03130_TR-eID-Server_Part1_pdf.pdf)
- 3819 [WSA] Web Services Addressing 1.0 - Core, W3C Recommendation 9 May 2006,  
3820 <http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/>
- 3821 [WSAM] Web Services Addressing 1.0 - Metadata, W3C Proposed Recommendation 31 July  
3822 2007, <http://www.w3.org/TR/ws-addr-metadata/>
- 3823 [WSASOAP] Web Services Addressing 1.0 – SOAP Binding, W3C Recommendation 9 May 2006,  
3824 <http://www.w3.org/TR/2006/REC-ws-addr-soap-20060509/>
- 3825 [WSAW] Web Services Addressing 1.0 – WSDL Binding, W3C Candidate Recommendation 29  
3826 May 2006, <http://www.w3.org/TR/ws-addr-wsdl/>
- 3827 [WSDL11] Web Services Description Language (WSDL) 1.1: W3C Note 15 March 2001,  
3828 <http://www.w3.org/TR/2001/NOTE-wsdl-20010315>
- 3829 [WSDL20] Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language,  
3830 W3C Recommendation 26 June 2007, <http://www.w3.org/TR/wsdl20/>
- 3831 [WSDLA] Web Services Description Language (WSDL) Version 2.0 Part 2: Adjuncts, W3C  
3832 Recommendation 26 June 2007, [http://www.w3.org/TR/2007/REC-wsdl20-](http://www.w3.org/TR/2007/REC-wsdl20-adjuncts-20070626/)  
3833 [adjuncts-20070626/](http://www.w3.org/TR/2007/REC-wsdl20-adjuncts-20070626/)
- 3834 [WSF] Web Services Federation Language (WS-Federation), Version 1.2, 22 May 2009,  
3835 [http://docs.oasis-open.org/wsfed/federation/v1.2/os/ws-](http://docs.oasis-open.org/wsfed/federation/v1.2/os/ws-federation-1.2-spec-os.pdf)  
3836 [federation-1.2-spec-os.pdf](http://docs.oasis-open.org/wsfed/federation/v1.2/os/ws-federation-1.2-spec-os.pdf)
- 3837 [WSI-Basic] WS-I Basic Profile 2.0, [Final Material, 2010-11-09](http://www.w3.org/2006/03/WSI-BasicProfile-2.0-2010-11-09), WEB SERVICES  
3838 INTEROPERABILITY ORGANIZATION, [http://ws-](http://www.w3.org/2006/03/WSI-BasicProfile-2.0-2010-11-09)  
3839 [i.org/Profiles/BasicProfile-2.0-2010-11-09.html](http://www.w3.org/2006/03/WSI-BasicProfile-2.0-2010-11-09)
- 3840 [WSI-BSP11] WS-I Basic Security Profile Version 1.1, Final Material, 2010-01-24, WEB SERVICES  
3841 INTEROPERABILITY ORGANIZATION, [http://www.ws-](http://www.w3.org/2006/03/WSI-BasicSecurityProfile-1.1)  
3842 [i.org/Profiles/BasicSecurityProfile-1.1.html](http://www.w3.org/2006/03/WSI-BasicSecurityProfile-1.1)
- 3843 [WSMC] Web Services Make Connection (WS MakeConnection), Version 1.0, OASIS  
3844 Standard, 14 June 2007, [http://docs.oasis-open.org/ws-](http://docs.oasis-open.org/ws-rx/wsmc/200702/wsmc-1.0-spec-os-01.pdf)  
3845 [rx/wsmc/200702/wsmc-1.0-spec-os-01.pdf](http://docs.oasis-open.org/ws-rx/wsmc/200702/wsmc-1.0-spec-os-01.pdf)
- 3846 [WSPA] Web Services Policy 1.5 - Attachment, W3C Recommendation, 4 September 2007;  
3847 <http://www.w3.org/TR/2007/REC-ws-policy-attach-20070904/>

- 3848 [WSPF] Web Services Policy 1.5 - Framework, W3C Recommendation, 4 September 2007;  
3849 <http://www.w3.org/TR/2007/REC-ws-policy-20070904/>
- 3850 [WSRM] Web Services Reliable Messaging (WS-ReliableMessaging) Version 1.1, OASIS  
3851 Standard Specification incorporating approved Errata, 07 January 2008,  
3852 <http://docs.oasis-open.org/ws-rx/wsrmp/200702/wsrmp-1.1-spec-os-01-e1.pdf>  
3853
- 3854 [WSRMP] Web Services Reliable Messaging Policy Assertion (WS-RM Policy) Version 1.1  
3855 OASIS Standard Specification incorporating approved Errata, 07 January 2008,  
3856 <http://docs.oasis-open.org/ws-rx/wsrmp/200702/wsrmp-1.1-spec-os-01-e1.pdf>  
3857
- 3858 [WSS] Web Services Security: SOAP Message Security Version 1.1.1, OASIS Standard, 18  
3859 May 2012 <http://docs.oasis-open.org/wss-m/wss/v1.1.1/wss-SOAPMessageSecurity-v1.1.1.pdf>  
3860
- 3861 [WSSC] Web Services Secure Conversation 1.3, OASIS Standard, 1 March 2007,  
3862 <http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/ws-secureconversation-1.3-os.pdf>  
3863
- 3864 [WSSKERB] Web Services Security Kerberos Token Profile 1.1.1, OASIS Standard, 18 May 2012,  
3865 <http://docs.oasis-open.org/wss-m/wss/v1.1.1/os/wss-KerberosTokenProfile-v1.1.1-os.pdf>  
3866
- 3867 [WSSP] WS-SecurityPolicy 1.3, OASIS Standard incorporating Approved Errata 01, 25 April  
3868 2012; <http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.3/errata01/os/ws-securitypolicy-1.3-errata01-os-complete.pdf>  
3869  
3870
- 3871 [WSSSAML] Web Services Security SAML Token Profile 1.1.1, OASIS Standard, 18 May 2012,  
3872 <http://docs.oasis-open.org/wss-m/wss/v1.1.1/os/wss-SAMLSecurityTokenProfile-v1.1.1-os.pdf>  
3873
- 3874 [WSSUSER] Web Services Security Username Token Profile 1.1.1, OASIS Standard, 18 May  
3875 2012, <http://docs.oasis-open.org/wss-m/wss/v1.1.1/os/wss-UsernameTokenProfile-v1.1.1-os.pdf>  
3876
- 3877 [WSSX509] Web Services Security X.509 Certificate Token Profile 1.1.1, OASIS Standard, , , 18  
3878 May 2012, <http://docs.oasis-open.org/wss-m/wss/v1.1.1/os/wss-x509TokenProfile-v1.1.1-os.pdf>  
3879
- 3880 [WST] WS-Trust 1.4, OASIS Standard incorporating Approved Errata 01, 25 April 2012;  
3881 <http://docs.oasis-open.org/ws-sx/ws-trust/v1.4/errata01/os/ws-trust-1.4-errata01-os-complete.pdf>  
3882
- 3883 [XAdES] European Telecommunications Standards Institute. ETSI TS 101 903: XML Advanced  
3884 Electronic Signatures, V1.4.2 2010-12;  
3885 [http://pda.etsi.org/exchangefolder/ts\\_101903v010402p.pdf](http://pda.etsi.org/exchangefolder/ts_101903v010402p.pdf)
- 3886 [XAdES-B] European Telecommunications Standards Institute. ETSI TS 103 171: XAdES  
3887 Baseline Profile, V2.1.1 2012-03;  
3888 [http://pda.etsi.org/exchangefolder/ts\\_103171v020101p.pdf](http://pda.etsi.org/exchangefolder/ts_103171v020101p.pdf)
- 3889 [XENC] World Wide Web Consortium. XML Encryption Syntax and Processing, W3C  
3890 Recommendation, 10.12.2002;  
3891 <http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/>
- 3892 [XKMS] XML Key Management Specification (XKMS 2.0) v2, W3C Recommendation, 28 June  
3893 2005, <http://www.w3.org/TR/2005/REC-xkms2-20050628/>
- 3894 [XKMSEU] PEPPOL XKMS v2 Interface Specification, Revision 2.2, PEPPOL WP1 2010-04-30,  
3895 [https://joinup.ec.europa.eu/svn/peppol/PEPPOL\\_EIA/1-](https://joinup.ec.europa.eu/svn/peppol/PEPPOL_EIA/1-)

- 3896 [ICT\\_Architecture/1-ICT-eSignature\\_Infrastructure/13-ICT-](#)  
3897 [Models/XKMS\\_Interface\\_Specification-220.pdf](#) (for correct presentation,  
3898 file must be downloaded and opened as PDF)
- 3899 [XMLDSIG] World Wide Web Consortium. XML-Signature Syntax and Processing (Second  
3900 Edition), W3C Recommendation, 10 June 2008;  
3901 <http://www.w3.org/TR/xmlsig-core/>
- 3902 [XMLSchema] World Wide Web Consortium. XML Schema, Parts 0, 1, and 2 (Second Edition). W3C  
3903 Recommendation, 28 October 2004; <http://www.w3.org/TR/xmlschema-0/>,  
3904 <http://www.w3.org/TR/xmlschema-1/>, and  
3905 <http://www.w3.org/TR/xmlschema-2/>
- 3906 [XML 1.0] World Wide Web Consortium. Extensible Markup Language (XML) 1.0 (Fourth  
3907 Edition), T. Bray, J. Paoli, C. M. Sperberg-McQueen, and E. Maler, Editors. 10  
3908 February 1998, revised 16 August 2006; [http://www.w3.org/TR/2006/REC-](http://www.w3.org/TR/2006/REC-xml-20060816/)  
3909 [xml-20060816/](http://www.w3.org/TR/2006/REC-xml-20060816/)
- 3910 [XOEVBH] IT Planungsrat, Handbuch zur Entwicklung XÖV-konformer Standards, Version 2.0, 1  
3911 August 2014, Section 8; [http://www.xoev.de/sixcms/media.php/13/XOEV-](http://www.xoev.de/sixcms/media.php/13/XOEV-Handbuch%202.pdf)  
3912 [Handbuch%202.pdf](http://www.xoev.de/sixcms/media.php/13/XOEV-Handbuch%202.pdf)
- 3913 [XPath 1.0] W3C Recommendation, "[XML Path Language \(XPath\) Version 1.0](#)," 16  
3914 November 1999; <http://www.w3.org/TR/xpath>
- 3915 **13.2 Informative**
- 3916 [WSFED] Web Services Federation Language (WS-Federation), Version 1.2, latest TC/Editor  
3917 Draft see: [http://www.oasis-](http://www.oasis-open.org/committees/documents.php?wg_abbrev=wsfed)  
3918 [open.org/committees/documents.php?wg\\_abbrev=wsfed](http://www.oasis-open.org/committees/documents.php?wg_abbrev=wsfed)
- 3919 [WSMEX] Web Services Metadata Exchange, Version 1.1, W3C Member Submission 13 August  
3920 2008, [http://www.w3.org/Submission/2008/SUBM-WS-](http://www.w3.org/Submission/2008/SUBM-WS-MetadataExchange-20080813/)  
3921 [MetadataExchange-20080813/](http://www.w3.org/Submission/2008/SUBM-WS-MetadataExchange-20080813/)

## 3922 Appendix A. Schema OSCI Transport 2.02

3923

3924

3925

3926

3927

3928

3929

3930

3931

3932

3933

3934

3935

3936

3937

3938

3939

3940

3941

3942

3943

3944

3945

3946

3947

3948

3949

3950

3951

3952

3953

3954

3955

3956

3957

3958

3959

3960

3961

3962

3963

3964

3965

3966

3967

3968

3969

3970

3971

3972

3973

3974

3975

3976

3977

3978

3979

3980

3981

3982

3983

3984

3985

3986

3987

```

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns:wsa="http://www.w3.org/2005/08/addressing"
xmlns:osci="http://www.osci.eu/ws/2008/05/transport"
xmlns:oscimeta="http://www.osci.eu/ws/2014/10/transport"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
utility-1.0.xsd" xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
xmlns:wsp="http://www.w3.org/ns/ws-policy"
targetNamespace="http://www.osci.eu/ws/2008/05/transport"
elementFormDefault="qualified" attributeFormDefault="unqualified">
  <!--OSCI Transport Version 2.02 schema - last edited 2015-01-27 -->
  <!--OSCI Transport 2.02 schema extended by metadata header for OSCI2.0,
  according modification for MsgBoxStatusList; MsgBoxFetchRequest attributed for
  requesting whole envelope, headers of body of original message only-->
  <!--xs:import namespace="http://www.osci.eu/ws/2014/10/transport"
  schemaLocation="http://www.osci.eu/ws/2014/10/transport/OSCI21_MessageMetaData_V2.0
  2.xsd"/-->
  <xs:import namespace="http://www.osci.eu/ws/2014/10/transport"
  schemaLocation="OSCI_MessageMetaData_V2.02.xsd"/>
  <xs:import namespace="http://www.w3.org/ns/ws-policy"
  schemaLocation="http://www.w3.org/2007/02/ws-policy.xsd"/>
  <xs:import namespace="http://www.w3.org/2005/08/addressing"
  schemaLocation="http://www.w3.org/2006/03/addressing/ws-addr.xsd"/>
  <xs:import namespace="http://www.w3.org/2000/09/xmldsig#"
  schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-schema.xsd"/>
  <xs:import namespace="http://www.w3.org/2003/05/soap-envelope"
  schemaLocation="http://www.w3.org/2003/05/soap-envelope"/>
  <xs:import namespace="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
  wssecurity-utility-1.0.xsd" schemaLocation="http://docs.oasis-
  open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"/>
  <!--WSA-Extension: BusinessScenarioType-->
  <xs:complexType name="TypeOfBusinessScenarioType">
    <xs:simpleContent>
      <xs:extension base="xs:anyURI">
        <xs:attribute ref="wsa:IsReferenceParameter" use="optional"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:element name="TypeOfBusinessScenario" type="osci:TypeOfBusinessScenarioType"/>
  <!--General header-part of OSCI messages: timestamps-->
  <xs:complexType name="MsgTimeStampsType">
    <xs:sequence>
      <xs:element name="ObsoleteAfter" type="xs:date" minOccurs="0">
        <xs:annotation>
          <xs:documentation>Date, when this message is obsolete; may be set by
  Initiator</xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="Delivery" type="xs:dateTime" minOccurs="0">
        <xs:annotation>
          <xs:documentation>Time of entry in a Recipient MsgBox</xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="InitialFetch" type="xs:dateTime" minOccurs="0">
        <xs:annotation>
          <xs:documentation>Time of first comitted fetch from MsgBox by the
  Recipient</xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="Reception" type="xs:dateTime" minOccurs="0">
        <xs:annotation>
          <xs:documentation>Reception Time set by the Recipient</xs:documentation>
        </xs:annotation>
      </xs:element>
    </xs:sequence>
  </xs:complexType>

```

```

3988     </xs:element>
3989   </xs:sequence>
3990 </xs:complexType>
3991 <xs:element name="MsgTimeStamps" type="osci:MsgTimeStampsType"/>
3992 <!--Types and Elements for MsgBox request/responses-->
3993 <xs:annotation>
3994   <xs:documentation>Template for MsgBox-Requests</xs:documentation>
3995 </xs:annotation>
3996 <xs:complexType name="MsgBoxRequestType">
3997   <xs:sequence>
3998     <xs:element ref="osci:MsgSelector" minOccurs="0"/>
3999   </xs:sequence>
4000 </xs:complexType>
4001 <xs:simpleType name="MsgBoxReasonEnum">
4002   <xs:restriction base="xs:anyURI">
4003     <xs:enumeration
4004       value="http://www.osci.eu/ws/2008/05/transport/MsgBox/reasons/NoMatch"/>
4005     <xs:enumeration
4006       value="http://www.osci.eu/ws/2008/05/transport/MsgBox/reasons/SearchArgsInvalid"/>
4007     <xs:enumeration
4008       value="http://www.osci.eu/ws/2008/05/transport/MsgBox/reasons/RequestIdInvalid"/>
4009     </xs:restriction>
4010 </xs:simpleType>
4011 <xs:simpleType name="MsgBoxReasonOpenEnum">
4012   <xs:union memberTypes="osci:MsgBoxReasonEnum xs:anyURI"/>
4013 </xs:simpleType>
4014 <xs:complexType name="MsgBoxResponseType">
4015   <xs:choice>
4016     <xs:element name="NoMessageAvailable">
4017       <xs:complexType>
4018         <xs:attribute name="reason" type="osci:MsgBoxReasonOpenEnum"
4019           use="required"/>
4020       </xs:complexType>
4021     </xs:element>
4022     <xs:element name="ItemsPending" type="xs:nonNegativeInteger"/>
4023   </xs:choice>
4024   <xs:attribute name="MsgBoxRequestID" type="xs:anyURI" use="required"/>
4025 </xs:complexType>
4026 <xs:complexType name="MsgAttributeListType">
4027   <xs:sequence>
4028     <xs:element ref="wsa:MessageID"/>
4029     <xs:element ref="wsa:RelatesTo" minOccurs="0" maxOccurs="unbounded"/>
4030     <xs:element ref="wsa:From" minOccurs="0"/>
4031     <xs:element ref="osci:TypeOfBusinessScenario"/>
4032     <xs:element name="MsgSize" type="xs:int"/>
4033     <!--xs:element ref="osci:MsgTimeStamps"/-->
4034     <xs:element name="ObsoleteAfterDate" type="xs:date" minOccurs="0"/>
4035     <xs:element name="DeliveryTime" type="xs:dateTime"/>
4036     <xs:element name="InitialFetchedTime" type="xs:dateTime" minOccurs="0"/>
4037   </xs:sequence>
4038 </xs:complexType>
4039 <xs:attribute name="MsgBoxRequestID" type="xs:anyURI"/>
4040 <xs:element name="MsgSelector">
4041   <xs:complexType>
4042     <xs:sequence minOccurs="0">
4043       <xs:element ref="wsa:MessageID" minOccurs="0" maxOccurs="unbounded"/>
4044       <xs:element ref="wsa:RelatesTo" minOccurs="0" maxOccurs="unbounded"/>
4045       <xs:element name="MsgBoxEntryTimeFrom" type="xs:dateTime" minOccurs="0"/>
4046       <xs:element name="MsgBoxEntryTimeTo" type="xs:dateTime" minOccurs="0"/>
4047       <xs:element name="Extension" type="xs:anyType" minOccurs="0"/>
4048     </xs:sequence>
4049     <xs:attribute name="newEntry" type="xs:boolean"/>
4050   </xs:complexType>
4051 </xs:element>
4052 <xs:element name="MsgStatusList" type="osci:MsgStatusListType"/>
4053 <xs:complexType name="MsgStatusListType">
4054   <xs:sequence>
4055     <xs:element name="MsgAttributes" type="osci:MsgAttributeListType" minOccurs="0"
4056       maxOccurs="unbounded"/>

```

```

4057     <xs:element ref="oscimeta:MessageMetaData" minOccurs="0"
4058 maxOccurs="unbounded"/>
4059   </xs:sequence>
4060 </xs:complexType>
4061 <xs:element name="MsgBoxFetchRequest">
4062   <xs:complexType>
4063     <xs:complexContent>
4064       <xs:extension base="osci:MsgBoxRequestType">
4065         <xs:attribute name="MsgPart" default="Envelope">
4066           <xs:simpleType>
4067             <xs:restriction base="xs:NMTOKEN">
4068               <xs:enumeration value="Envelope"/>
4069               <xs:enumeration value="Header"/>
4070               <xs:enumeration value="Body"/>
4071             </xs:restriction>
4072           </xs:simpleType>
4073         </xs:attribute>
4074       </xs:extension>
4075     </xs:complexContent>
4076   </xs:complexType>
4077 </xs:element>
4078 <xs:element name="MsgBoxStatusListRequest"
4079 type="osci:MsgBoxStatusListRequestType"/>
4080 <xs:complexType name="MsgBoxStatusListRequestType">
4081   <xs:complexContent>
4082     <xs:extension base="osci:MsgBoxRequestType">
4083       <xs:attribute name="maxListItems" type="xs:positiveInteger"/>
4084       <xs:attribute name="ListForm">
4085         <xs:simpleType>
4086           <xs:restriction base="xs:NMTOKEN">
4087             <xs:enumeration value="MsgAttributes"/>
4088             <xs:enumeration value="MessageMetaData"/>
4089           </xs:restriction>
4090         </xs:simpleType>
4091       </xs:attribute>
4092     </xs:extension>
4093   </xs:complexContent>
4094 </xs:complexType>
4095 <xs:element name="MsgBoxResponse" type="osci:MsgBoxResponseType"/>
4096 <xs:element name="MsgBoxGetNextRequest" type="osci:MsgBoxGetNextRequestType"/>
4097 <xs:complexType name="MsgBoxGetNextRequestType">
4098   <xs:sequence minOccurs="0">
4099     <xs:element name="LastMsgReceived" type="wsa:AttributedURIType"
4100 maxOccurs="unbounded"/>
4101   </xs:sequence>
4102   <xs:attribute name="MsgBoxRequestID" type="xs:anyURI" use="required"/>
4103 </xs:complexType>
4104 <xs:element name="MsgBoxCloseRequest" type="osci:MsgBoxCloseRequestType"/>
4105 <xs:complexType name="MsgBoxCloseRequestType">
4106   <xs:sequence minOccurs="0">
4107     <xs:element name="LastMsgReceived" type="wsa:AttributedURIType"
4108 maxOccurs="unbounded"/>
4109   </xs:sequence>
4110   <xs:attribute name="MsgBoxRequestID" type="xs:anyURI" use="required"/>
4111 </xs:complexType>
4112 <!--Types and Elements for Receipt- and Notification Handling-->
4113 <xs:attribute name="qualTSPForReceipt" type="xs:boolean" default="false"/>
4114 <xs:attribute name="echoRequest" type="xs:boolean" default="false"/>
4115 <xs:complexType name="ReceiptDemandType">
4116   <xs:sequence>
4117     <xs:element ref="wsa:ReplyTo"/>
4118   </xs:sequence>
4119   <xs:attribute name="qualTSPForReceipt" type="xs:boolean" default="false"/>
4120   <xs:attribute name="echoRequest" type="xs:boolean" default="false"/>
4121 </xs:complexType>
4122 <xs:element name="DeliveryReceiptDemand" type="osci:DeliveryReceiptDemandType"/>
4123 <xs:element name="ReceptionReceiptDemand" type="osci:ReceptionReceiptDemandType"/>
4124 <xs:element name="ReceiptInfo" type="osci:ReceiptInfoType"/>
4125 <xs:complexType name="ReceiptInfoType">

```

```

4126 <xs:sequence>
4127   <xs:element ref="wsa:MessageID"/>
4128   <xs:element ref="osci:MsgTimeStamps"/>
4129   <xs:element ref="wsa:RelatesTo" minOccurs="0" maxOccurs="unbounded"/>
4130   <xs:element name="To" type="wsa:EndpointReferenceType"/>
4131   <xs:element ref="wsa:From" minOccurs="0"/>
4132   <xs:element ref="wsa:ReplyTo"/>
4133   <xs:element name="RequestEcho" type="xs:base64Binary" minOccurs="0"/>
4134   <xs:element ref="oscimeta:MessageMetaData" minOccurs="0"/>
4135 </xs:sequence>
4136 <xs:attribute name="Id" type="xs:ID" use="required"/>
4137 <xs:attribute name="ReceiptIssuerRole" use="optional">
4138   <xs:simpleType>
4139     <xs:restriction base="xs:anyURI">
4140       <xs:enumeration
4141 value="http://www.osci.eu/ws/2008/05/transport/role/MsgBox"/>
4142       <xs:enumeration
4143 value="http://www.osci.eu/ws/2008/05/transport/role/Recipient"/>
4144       <xs:enumeration value="http://www.osci.eu/ws/2008/05/transport/role/Sender
4145 "/>
4146       <xs:enumeration value="http://www.osci.eu/ws/2008/05/transport/role/Relay
4147 "/>
4148     </xs:restriction>
4149   </xs:simpleType>
4150 </xs:attribute>
4151 </xs:complexType>
4152 <xs:complexType name="DeliveryReceiptDemandType">
4153   <xs:complexContent>
4154     <xs:restriction base="osci:ReceiptDemandType">
4155       <xs:sequence>
4156         <xs:element ref="wsa:ReplyTo"/>
4157       </xs:sequence>
4158     </xs:restriction>
4159   </xs:complexContent>
4160 </xs:complexType>
4161 <xs:complexType name="ReceptionReceiptDemandType">
4162   <xs:complexContent>
4163     <xs:restriction base="osci:ReceiptDemandType">
4164       <xs:sequence>
4165         <xs:element ref="wsa:ReplyTo"/>
4166       </xs:sequence>
4167     </xs:restriction>
4168     <!-- xs:attribute ref="s12:role" fixed="http://www.w3.org/2003/05/soap-
4169 envelope/role/ultimateReceiver"/-->
4170   </xs:complexContent>
4171 </xs:complexType>
4172 <xs:complexType name="DeliveryReceiptType">
4173   <xs:sequence>
4174     <xs:element ref="osci:ReceiptInfo"/>
4175     <xs:element ref="ds:Signature"/>
4176   </xs:sequence>
4177 </xs:complexType>
4178 <xs:element name="DeliveryReceipt" type="osci:DeliveryReceiptType"/>
4179 <xs:element name="SubmissionReceipt" type="osci:DeliveryReceiptType"/>
4180 <xs:element name="RelayReceipt" type="osci:DeliveryReceiptType"/>
4181 <xs:complexType name="ReceptionReceiptType">
4182   <xs:sequence>
4183     <xs:element ref="osci:ReceiptInfo"/>
4184     <xs:element ref="ds:Signature"/>
4185   </xs:sequence>
4186 </xs:complexType>
4187 <xs:element name="ReceptionReceipt" type="osci:ReceptionReceiptType"/>
4188 <xs:complexType name="FetchedNotificationDemandType">
4189   <xs:sequence>
4190     <xs:element ref="wsa:ReplyTo"/>
4191   </xs:sequence>
4192   <xs:attribute ref="s12:role"
4193 default="http://www.osci.eu/ws/2008/05/transport/role/MsgBox"/>
4194 </xs:complexType>

```

```

4195 <xs:element name="FetchedNotificationDemand"
4196 type="osci:FetchedNotificationDemandType"/>
4197 <xs:complexType name="FetchedNotificationType">
4198 <xs:sequence>
4199 <xs:element name="FetchedTime" type="xs:dateTime"/>
4200 <xs:element ref="wsa:MessageID"/>
4201 <xs:element ref="wsa:To"/>
4202 <xs:element ref="wsa:From"/>
4203 </xs:sequence>
4204 </xs:complexType>
4205 <xs:element name="FetchedNotification" type="osci:FetchedNotificationType"/>
4206 <!--Extensions for Key usage context-->
4207 <xs:complexType name="X509TokenContainerType">
4208 <xs:sequence maxOccurs="unbounded">
4209 <xs:element ref="osci:X509TokenInfo"/>
4210 </xs:sequence>
4211 <xs:attribute name="validateCompleted" type="xs:boolean" default="false"/>
4212 </xs:complexType>
4213 <xs:element name="X509TokenContainer" type="osci:X509TokenContainerType"/>
4214 <xs:element name="X509TokenInfo">
4215 <xs:complexType>
4216 <xs:sequence>
4217 <xs:element ref="ds:X509Data"/>
4218 <xs:element name="TokenApplication" maxOccurs="unbounded">
4219 <xs:complexType>
4220 <xs:sequence>
4221 <xs:element name="TimeInstant" type="xs:dateTime"/>
4222 <xs:element name="MsgItemRef" type="xs:IDREF" minOccurs="0"/>
4223 </xs:sequence>
4224 <xs:attribute name="validateResultRef" type="xs:IDREF"/>
4225 <xs:attribute name="ocspNoCache" type="xs:boolean"/>
4226 </xs:complexType>
4227 </xs:element>
4228 </xs:sequence>
4229 <xs:attribute name="validated" type="xs:boolean" default="false"/>
4230 <xs:attribute name="Id" type="xs:ID" use="required"/>
4231 <!-- RFC 3280 for KeyUsage with Extentensions Attribute Certificate and usage
4232 for Authentication -->
4233 </xs:complexType>
4234 <!--OSCI Policy Asserstions-->
4235 <!--Policy qualified Timestamp Servcie available-->
4236 </xs:element>
4237 <!--Poliy Assertion carrying Endpoints X509Certificates-->
4238 <xs:element name="X509CertificateAssertion">
4239 <xs:complexType>
4240 <xs:sequence>
4241 <xs:element ref="wsp:All"/>
4242 </xs:sequence>
4243 </xs:complexType>
4244 </xs:element>
4245 <!--Policy, when qualified TSP service can be requested from this node-->
4246 <xs:element name="QualTspAssertion">
4247 <xs:complexType>
4248 <xs:attribute name="PolicyRef" type="xs:anyURI"/>
4249 </xs:complexType>
4250 </xs:element>
4251 <!--Policy if and how MsgTimeStamps:OsoleteAfter is handled-->
4252 <xs:element name="ObsoleteAfterAssertion">
4253 <xs:complexType>
4254 <xs:sequence>
4255 <xs:element name="MsgRetainDays" type="xs:positiveInteger"/>
4256 <xs:element name="WarningBeforeMsgObsolete" type="xs:positiveInteger"
4257 minOccurs="0"/>
4258 </xs:sequence>
4259 <xs:attribute name="PolicyRef" type="xs:anyURI"/>
4260 </xs:complexType>
4261 </xs:element>
4262 <!--Poliy for MakeConnection: Response Retention Days-->
4263 <xs:element name="MsgRetainDays" type="xs:positiveInteger"/>

```

```
4264 <!--Enumeration for possible X509 Token Usages-->
4265 <xs:attribute name="TokenUsage">
4266   <xs:simpleType>
4267     <xs:restriction base="xs:anyURI">
4268       <xs:enumeration
4269 value="http://www.osci.eu/common/names/TokenUsage/e2eContentEncryption"/>
4270       <xs:enumeration
4271 value="http://www.osci.eu/common/names/TokenUsage/TransportEncryption"/>
4272       <xs:enumeration
4273 value="http://www.osci.eu/common/names/TokenUsage/ReceiptSigning"/>
4274       <xs:enumeration
4275 value="http://www.osci.eu/common/names/TokenUsage/TSPSigning"/>
4276     </xs:restriction>
4277   </xs:simpleType>
4278 </xs:attribute>
4279 <!--Opaque Body Type - not used-->
4280 <!--Policy maximum accepted Message size and Frequency per hour-->
4281 <xs:element name="AcceptedMsgLimits">
4282   <xs:complexType>
4283     <xs:sequence>
4284       <xs:element name="MaxSize" type="xs:positiveInteger"/>
4285       <xs:element name="MaxPerHour" type="xs:positiveInteger"/>
4286     </xs:sequence>
4287   </xs:complexType>
4288 </xs:element>
4289 <xs:complexType name="MessageBody">
4290   <xs:sequence>
4291     <xs:any namespace="##any" minOccurs="0" maxOccurs="unbounded"/>
4292   </xs:sequence>
4293 </xs:complexType>
4294 </xs:schema>
```

4295

## Appendix B. OSCI Transport 2.02 – Schema MessageMetaData

4296

4297

4298

4299

4300

4301

4302

4303

4304

4305

4306

4307

4308

4309

4310

4311

4312

4313

4314

4315

4316

4317

4318

4319

4320

4321

4322

4323

4324

4325

4326

4327

4328

4329

4330

4331

4332

4333

4334

4335

4336

4337

4338

4339

4340

4341

4342

4343

4344

4345

4346

4347

4348

4349

4350

4351

4352

4353

4354

4355

4356

4357

4358

4359

```

<?xml version="1.0" encoding="UTF-8"?>
<!--Schema for OSCI Message Meta Data - last edited 2015-02-19 -->
<!-- Change 2015-02-19: MessageType amended by mandatory attribte @payloadSchema --
>
<!-- Change 2015-01-23: Alignment with XTA/KoSIT: introduced KeyCodeType, changed
PropertyType, BusinessScenarioType, MessageType; ServiceQuality (to #any type),
SecurityToken may carry IDREF attribute to token in payload now; usage attribute
mandatory now -->
<!-- Change 2014-11-30: xoev basis data type schema version changed from 1_0 to 1_1
-->
<!-- Last recent changes: Codelist for BusinessScenarioTypes defined and imported -
->
<!-- Changes: 2.0.2: Adoption of xoev:Codelist type for some elements; eliminating
QName typed attributes/elements; PartyType elements now may include optional
SecurityTokens (as e.g. used in XVergabe) -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:oscimeta="http://www.osci.eu/ws/2014/10/transport"
xmlns:wsa="http://www.w3.org/2005/08/addressing" xmlns:wsse="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:xoev-
dt="http://xoev.de/schemata/basisdatentypen/1_1"
targetNamespace="http://www.osci.eu/ws/2014/10/transport"
elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:import namespace="http://www.w3.org/2005/08/addressing"
schemaLocation="http://www.w3.org/2006/03/addressing/ws-addr.xsd"/>
  <xs:import namespace="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd" schemaLocation="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"/>
  <xs:import namespace="http://xoev.de/schemata/basisdatentypen/1_1"
schemaLocation="http://xoev.de/schemata/basisdatentypen/1_1/xoev-
basisdatentypen.xsd"/>
  <xs:simpleType name="NonEmptyStringType">
    <xs:restriction base="xs:string">
      <xs:minLength value="1"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="NonEmptyURIType">
    <xs:restriction base="xs:anyURI">
      <xs:minLength value="1"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="AnyType" mixed="true">
    <xs:sequence minOccurs="0" maxOccurs="unbounded">
      <xs:any namespace="##any" processContents="lax"/>
    </xs:sequence>
    <xs:anyAttribute namespace="##any"/>
  </xs:complexType>
  <!-- End AnyType -->
  <xs:complexType name="ReceiptRequestType">
    <xs:sequence>
      <xs:element name="Submission" minOccurs="0">
        <xs:annotation>
          <xs:documentation>Sending node: Message accepted for delivery and
submitted</xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="Relay" minOccurs="0">
        <xs:annotation>
          <xs:documentation>Active node on the delivery route: Message forwarded to
next hop</xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="Delivery" minOccurs="0">

```

```

4360     <xs:annotation>
4361     <xs:documentation>Destination node:Successful delivery to Recipient in
4362     synchronous scenarios, to MsgBox if asynchronous</xs:documentation>
4363     </xs:annotation>
4364     </xs:element>
4365     <xs:element name="Fetch" minOccurs="0">
4366     <xs:annotation>
4367     <xs:documentation>Only MsgBox node: Initial fetch of Message by Recipient
4368     from his MsgBox</xs:documentation>
4369     </xs:annotation>
4370     </xs:element>
4371     <xs:element name="Reception" minOccurs="0">
4372     <xs:annotation>
4373     <xs:documentation>Ultimate Recipient node, after acceptance of message,
4374     after successful decryption of payload</xs:documentation>
4375     </xs:annotation>
4376     </xs:element>
4377     <xs:element name="ReceiptTo" type="wsa:EndpointReferenceType" minOccurs="0"/>
4378     </xs:sequence>
4379   </xs:complexType>
4380   <xs:complexType name="DeliveryAttributesType">
4381     <xs:annotation>
4382     <xs:documentation>Message delivery time instants, quality and receipts
4383     requested</xs:documentation>
4384     </xs:annotation>
4385     <xs:sequence>
4386     <xs:annotation>
4387     <xs:documentation>Timestamps, priority etc.</xs:documentation>
4388     </xs:annotation>
4389     <xs:element name="Origin" type="xs:dateTime" minOccurs="0">
4390     <xs:annotation>
4391     <xs:documentation>Production of content by Requester respective (response)
4392     Provider</xs:documentation>
4393     </xs:annotation>
4394     </xs:element>
4395     <xs:element name="InitialSend" type="xs:dateTime" minOccurs="0">
4396     <xs:annotation>
4397     <xs:documentation>Time when delivery was started (submission by Senders
4398     node)</xs:documentation>
4399     </xs:annotation>
4400     </xs:element>
4401     <xs:element name="NotBefore" type="xs:dateTime" minOccurs="0">
4402     <xs:annotation>
4403     <xs:documentation>Time when sending node should submit
4404     message</xs:documentation>
4405     </xs:annotation>
4406     </xs:element>
4407     <xs:element name="ObsoleteAfter" type="xs:date" minOccurs="0">
4408     <xs:annotation>
4409     <xs:documentation>Date, when this message is obsolete; may be set by
4410     Initiator</xs:documentation>
4411     </xs:annotation>
4412     </xs:element>
4413     <xs:element name="Delivery" type="xs:dateTime" minOccurs="0">
4414     <xs:annotation>
4415     <xs:documentation>Time of entry in a Recipients MsgBox or reception by
4416     Recipient in synchronous case</xs:documentation>
4417     </xs:annotation>
4418     </xs:element>
4419     <xs:element name="InitialFetch" type="xs:dateTime" minOccurs="0">
4420     <xs:annotation>
4421     <xs:documentation>Time of first comitted fetch from MsgBox by
4422     recipient</xs:documentation>
4423     </xs:annotation>
4424     </xs:element>
4425     <xs:element name="Reception" type="xs:dateTime" minOccurs="0">
4426     <xs:annotation>
4427     <xs:documentation>Reception time set by the Ultimate Recipient ("Reader",
4428     target application)</xs:documentation>

```

```

4429     </xs:annotation>
4430   </xs:element>
4431   <xs:element name="ServiceQuality" type="oscimeta:NonEmptyStringType"
4432   minOccurs="0">
4433     <xs:annotation>
4434       <xs:documentation>Property like priority etc. - XTA here points to "Service
4435 Profile </xs:documentation>
4436     </xs:annotation>
4437   </xs:element>
4438   <xs:element name="ReceiptRequests" type="oscimeta:ReceiptRequestType"
4439   minOccurs="0">
4440     <xs:annotation>
4441       <xs:documentation>Receipts requested by sender or author</xs:documentation>
4442     </xs:annotation>
4443   </xs:element>
4444 </xs:sequence>
4445 </xs:complexType>
4446 <xs:element name="SecurityToken">
4447   <xs:complexType>
4448     <xs:choice>
4449       <xs:element ref="wsse:BinarySecurityToken"/>
4450       <xs:element ref="wsse:SecurityTokenReference"/>
4451       <xs:element ref="wsse:UsernameToken"/>
4452     </xs:choice>
4453     <xs:attribute name="usage" use="required">
4454       <xs:simpleType>
4455         <xs:restriction base="xs:NMTOKEN">
4456           <xs:enumeration value="AUTHENTICATION"/>
4457           <xs:enumeration value="ENCRYPTION"/>
4458           <xs:enumeration value="SIGNATURE"/>
4459         </xs:restriction>
4460       </xs:simpleType>
4461     </xs:attribute>
4462     <xs:attribute name="payloadRef" type="xs:IDREF"/>
4463   </xs:complexType>
4464 </xs:element>
4465 <xs:complexType name="PartyType">
4466   <xs:annotation>
4467     <xs:documentation>Logical identifier and optional security tokens of that
4468 entity (binary, may carry SAML, too) </xs:documentation>
4469   </xs:annotation>
4470   <xs:sequence>
4471     <xs:element name="Identifier" type="oscimeta:PartyIdentifierType"/>
4472     <xs:element ref="oscimeta:SecurityToken" minOccurs="0" maxOccurs="unbounded"/>
4473   </xs:sequence>
4474 </xs:complexType>
4475 <xs:complexType name="PartyIdentifierType">
4476   <xs:annotation>
4477     <xs:documentation>Value of generic party identifier, as classified by @type
4478 attribute, e.g.: Prefix:Kennung</xs:documentation>
4479   </xs:annotation>
4480   <xs:simpleContent>
4481     <xs:extension base="xs:normalizedString">
4482       <xs:attribute name="type" type="oscimeta:NonEmptyStringType" use="required">
4483         <xs:annotation>
4484           <xs:documentation>Orientation: ebMS Core: type, how to interpret Party-Id
4485 value, e.g.: xöv oder Justiz</xs:documentation>
4486         </xs:annotation>
4487       </xs:attribute>
4488       <xs:attribute name="name" type="oscimeta:NonEmptyStringType">
4489         <xs:annotation>
4490           <xs:documentation>optional "friendly name" value for displaying in user
4491 agents (as e.g. known from eMail)</xs:documentation>
4492         </xs:annotation>
4493       </xs:attribute>
4494       <xs:attribute name="category" type="oscimeta:NonEmptyStringType">
4495         <xs:annotation>
4496           <xs:documentation>Concrete role of party in business scenario (e.g.
4497 "buyer", "Meldebehörde", "Standesamt"...)</xs:documentation>

```

```

4498     </xs:annotation>
4499     </xs:attribute>
4500   </xs:extension>
4501   </xs:simpleContent>
4502 </xs:complexType>
4503 <xs:element name="Author" type="oscimeta:PartyType">
4504   <xs:annotation>
4505     <xs:documentation>Requester resp. (response-) Provider</xs:documentation>
4506   </xs:annotation>
4507 </xs:element>
4508 <xs:element name="Reader" type="oscimeta:PartyType">
4509   <xs:annotation>
4510     <xs:documentation>Destinations of the message</xs:documentation>
4511   </xs:annotation>
4512 </xs:element>
4513 <xs:complexType name="OriginatorsType">
4514   <xs:sequence>
4515     <xs:element ref="oscimeta:Author"/>
4516     <xs:element ref="oscimeta:Sender" minOccurs="0"/>
4517     <xs:element name="ReplyTo" type="oscimeta:PartyType" minOccurs="0">
4518       <xs:annotation>
4519         <xs:documentation>If response expected different from value outlined in
4520 "From" address</xs:documentation>
4521       </xs:annotation>
4522     </xs:element>
4523   </xs:sequence>
4524 </xs:complexType>
4525 <xs:complexType name="DestinationsType">
4526   <xs:sequence>
4527     <xs:element ref="oscimeta:Reader">
4528       <xs:annotation>
4529         <xs:documentation>Ultimate target of the message</xs:documentation>
4530       </xs:annotation>
4531     </xs:element>
4532     <xs:element ref="oscimeta:OtherDestinations" minOccurs="0"/>
4533   </xs:sequence>
4534 </xs:complexType>
4535 <xs:complexType name="ProcessIdentifierType">
4536   <xs:annotation>
4537     <xs:documentation>Process ID message is related to</xs:documentation>
4538   </xs:annotation>
4539   <xs:simpleContent>
4540     <xs:extension base="oscimeta:NonEmptyStringType">
4541       <xs:attribute name="ProccesName" type="oscimeta:NonEmptyStringType">
4542         <xs:annotation>
4543           <xs:documentation>Process may have a name, e.g. "order"</xs:documentation>
4544         </xs:annotation>
4545       </xs:attribute>
4546     </xs:extension>
4547   </xs:simpleContent>
4548 </xs:complexType>
4549 <xs:complexType name="MsgIdentificationType">
4550   <xs:sequence>
4551     <xs:element ref="wsa:MessageID"/>
4552     <xs:element name="In-Reply-To" type="wsa:AttributedURIType" minOccurs="0"
4553 maxOccurs="unbounded">
4554       <xs:annotation>
4555         <xs:documentation>Referenced application level Message-
4556 Id(s)</xs:documentation>
4557       </xs:annotation>
4558     </xs:element>
4559     <xs:element name="ProcessRef" minOccurs="0">
4560       <xs:annotation>
4561         <xs:documentation>References to business process-id's (like ebMS
4562 Conversation-Id, "AktENZEICHEN" in Germany)</xs:documentation>
4563       </xs:annotation>
4564     </xs:complexType>
4565   </xs:sequence>

```

```

4566     <xs:element name="Requester" type="oscimeta:ProcessIdentifierType"
4567 minOccurs="0">
4568     <xs:annotation>
4569     <xs:documentation>Ref on requester (Source Application)
4570 side</xs:documentation>
4571     </xs:annotation>
4572     </xs:element>
4573     <xs:element name="Responder" type="oscimeta:ProcessIdentifierType"
4574 minOccurs="0">
4575     <xs:annotation>
4576     <xs:documentation>Ref on responder (Target Application) side, if
4577 different</xs:documentation>
4578     </xs:annotation>
4579     </xs:element>
4580     </xs:sequence>
4581     </xs:complexType>
4582     </xs:element>
4583     </xs:sequence>
4584 </xs:complexType>
4585 <xs:complexType name="KeyCodeType">
4586     <xs:complexContent>
4587     <xs:restriction base="xoev-dt:Code">
4588     <xs:sequence>
4589     <xs:element name="code" type="xs:token" form="unqualified"/>
4590     <xs:element name="name" type="xs:normalizedString" form="unqualified"
4591 minOccurs="0"/>
4592     </xs:sequence>
4593     <xs:attribute name="listURI" type="xs:anyURI" use="required"/>
4594     <xs:attribute name="listVersionID" type="xs:normalizedString"
4595 use="required"/>
4596     </xs:restriction>
4597     </xs:complexContent>
4598 </xs:complexType>
4599 <xs:complexType name="PropertyType">
4600     <xs:sequence>
4601     <xs:element name="Key" type="oscimeta:KeyCodeType"/>
4602     <xs:element name="Value" type="oscimeta:NonEmptyStringType"/>
4603     </xs:sequence>
4604 </xs:complexType>
4605 <xs:complexType name="MessagePropertiesType">
4606     <xs:sequence>
4607     <xs:element name="Property" type="oscimeta:PropertyType"
4608 maxOccurs="unbounded"/>
4609     </xs:sequence>
4610 </xs:complexType>
4611 <xs:complexType name="QualifierType">
4612     <xs:sequence>
4613     <xs:element name="Subject" type="xs:string" minOccurs="0">
4614     <xs:annotation>
4615     <xs:documentation>Message subject text (informational)</xs:documentation>
4616     </xs:annotation>
4617     </xs:element>
4618     <xs:element name="Service" type="xs:anyURI">
4619     <xs:annotation>
4620     <xs:documentation>Distinct service in a certain business scenario context;
4621 in the XÖV context this is the "Dienste URI"</xs:documentation>
4622     </xs:annotation>
4623     </xs:element>
4624     <xs:element name="BusinessScenario">
4625     <xs:annotation>
4626     <xs:documentation>Domain qualifier, e.g. Meldewesen,
4627 XVergabe...</xs:documentation>
4628     </xs:annotation>
4629     </xs:complexType>
4630     <xs:choice>
4631     <xs:element name="Defined" type="oscimeta:KeyCodeType"/>
4632     <xs:element name="Undefined" type="xs:normalizedString"/>
4633     </xs:choice>
4634     </xs:complexType>

```

```

4635     </xs:element>
4636     <xs:element name="MessageType">
4637         <xs:annotation>
4638             <xs:documentation>Payload: Type of document or message. MessageTypes
4639 normally bound to specific BusinessScenario</xs:documentation>
4640         </xs:annotation>
4641         <xs:complexType>
4642             <xs:complexContent>
4643                 <xs:extension base="oscimeta:KeyCodeType">
4644                     <xs:attribute name="payloadSchema" type="oscimeta:NonEmptyURIType"
4645 use="required"/>
4646                 </xs:extension>
4647             </xs:complexContent>
4648         </xs:complexType>
4649     </xs:element>
4650 </xs:sequence>
4651 </xs:complexType>
4652 <xs:element name="DeliveryAttributes" type="oscimeta:DeliveryAttributesType">
4653     <xs:annotation>
4654         <xs:documentation>Time stamps, receipts to be generated, service
4655 quality</xs:documentation>
4656     </xs:annotation>
4657 </xs:element>
4658 <xs:element name="Originators" type="oscimeta:OriginatorsType">
4659     <xs:annotation>
4660         <xs:documentation>Message originators and reply address</xs:documentation>
4661     </xs:annotation>
4662 </xs:element>
4663 <xs:element name="Destinations" type="oscimeta:DestinationsType">
4664     <xs:annotation>
4665         <xs:documentation>Actual and other destinations of Message</xs:documentation>
4666     </xs:annotation>
4667 </xs:element>
4668 <xs:element name="MsgIdentification" type="oscimeta:MsgIdentificationType">
4669     <xs:annotation>
4670         <xs:documentation>Message ID and Message relations</xs:documentation>
4671     </xs:annotation>
4672 </xs:element>
4673 <xs:element name="Qualifier" type="oscimeta:QualifierType">
4674     <xs:annotation>
4675         <xs:documentation>General payload properties, common to all
4676 scenarios</xs:documentation>
4677     </xs:annotation>
4678 </xs:element>
4679 <xs:element name="MessageProperties">
4680     <xs:annotation>
4681         <xs:documentation>Scenarios specific payload properties, to be agreed upon per
4682 scenario</xs:documentation>
4683     </xs:annotation>
4684     <xs:complexType>
4685         <xs:sequence>
4686             <xs:element name="Property" type="oscimeta:PropertyType"
4687 maxOccurs="unbounded"/>
4688         </xs:sequence>
4689     </xs:complexType>
4690 </xs:element>
4691 <xs:element name="Sender" type="oscimeta:PartyType">
4692     <xs:annotation>
4693         <xs:documentation>Sending node, entry may be added by Sender
4694 node</xs:documentation>
4695     </xs:annotation>
4696 </xs:element>
4697 <xs:element name="OtherDestinations">
4698     <xs:annotation>
4699         <xs:documentation>Other destinations of message - informational, as known from
4700 e-mail</xs:documentation>
4701     </xs:annotation>
4702     <xs:complexType>
4703         <xs:sequence>

```

```
4704     <xs:element ref="oscimeta:OtherReaders" maxOccurs="unbounded"/>
4705     <xs:element ref="oscimeta:CcReaders" minOccurs="0" maxOccurs="unbounded"/>
4706   </xs:sequence>
4707 </xs:complexType>
4708 </xs:element>
4709 <xs:element name="OtherReaders" type="oscimeta:PartyIdentifierType"/>
4710 <xs:element name="CcReaders" type="oscimeta:PartyIdentifierType">
4711   <xs:annotation>
4712     <xs:documentation>Destinations in cc role</xs:documentation>
4713   </xs:annotation>
4714 </xs:element>
4715 <xs:element name="MessageMetaData">
4716   <xs:complexType>
4717     <xs:sequence>
4718       <xs:element ref="oscimeta:DeliveryAttributes"/>
4719       <xs:element ref="oscimeta:Originators"/>
4720       <xs:element ref="oscimeta:Destinations"/>
4721       <xs:element ref="oscimeta:MsgIdentification"/>
4722       <xs:element ref="oscimeta:Qualifier"/>
4723       <xs:element ref="oscimeta:MessageProperties" minOccurs="0"/>
4724       <xs:element name="MsgSize" type="xs:positiveInteger" minOccurs="0">
4725         <xs:annotation>
4726           <xs:documentation>Message size in bytes</xs:documentation>
4727         </xs:annotation>
4728       </xs:element>
4729     </xs:sequence>
4730     <xs:attribute name="TestMsg" type="xs:boolean" default="false">
4731       <xs:annotation>
4732         <xs:documentation>"true", if test-message; defaults to
4733 "false"</xs:documentation>
4734       </xs:annotation>
4735     </xs:attribute>
4736   </xs:complexType>
4737 </xs:element>
4738 </xs:schema>
```

## 4739 Appendix C. Example: OSCI Endpoint Metadata 4740 Instance

4741 This example is a policy instance of the schema explained in chapter [10.2]. The encryption and  
4742 signature certificates exposed in this policy are addressed by URI references; according to [WSS]  
4743 they could also be embedded in this policy file itself in base64Binary format.

4744 This endpoint exposes different encryption and signature certificates for the MsgBox and recipient /  
4745 UltimateRecipient instances. The [ObsoleteAfter] property is handled by this endpoint, while a service  
4746 for qualified timestamps is not offered.

4747 For readability of policies used in the OSCI Transport context, it is strongly RECOMMENDED to  
4748 generally use the `wsu:id` attribute values highlighted **bold** in this example, as these policy parts and  
4749 certificates are referenced by other policies or service instances like a STS (i.e., the latter needs  
4750 access to the endpoint encryption certificate for appropriate SAML token encryption).

4751

```

4752 <?xml version="1.0" encoding="UTF-8" ?>
4753 <!-- OSCI specific endpoint metadata / policies -->
4754 <wsp:Policy Name="ExampleEndpointOSCIPolicy"
4755 xsi:schemaLocation="http://schemas.xmlsoap.org/ws/2004/09/policy
4756 http://schemas.xmlsoap.org/ws/2004/09/policy/wspolicy.xsd"
4757 xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
4758 utility-1.0.xsd" xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
4759 xmlns:wspimport="http://schemas.xmlsoap.org/ws/2004/09/policy/optionalimport"
4760 xmlns:wse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
4761 wssecurity-secext-1.0.xsd"
4762 xmlns:firmac="urn:de:egov:names:firm1.0:authenticate"
4763 xmlns:osci="http://www.osci.eu/ws/2008/05/transport.xsd"
4764 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
4765   <wsp:All>
4766
4767     <wsp:Policy wsu:id="X509CertificateAssertion">
4768       <osci:X509CertificateAssertion>
4769         <wsp:All>
4770           <wse:SecurityTokenReference wsu:id="UltimateRecipientEncCert"
4771 wse:Usage="http://www.osci.eu/2008/05/common/names/TokenUsage/e2eContentEncryption"
4772   osci:Role="http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver">
4773             <wse:Reference URI="REPLACE_WITH_ACTUAL_URL_to_UltimateRecipientEncCert"
4774 ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-
4775 profile-1.0#X509v3"/>
4776           </wse:SecurityTokenReference>
4777           <wse:SecurityTokenReference wsu:id="RecipientSignatureCert"
4778 wse:Usage="http://www.osci.eu/2008/05/common/names/TokenUsage/RecipientSignature"
4779   osci:Role="http://www.osci.eu/ws/2008/05/transport/role/Recipient">
4780             <wse:Reference URI="REPLACE_WITH_ACTUAL_URL_to_RecipientSignatureCert"
4781 ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-
4782 profile-1.0#X509v3"/>
4783           </wse:SecurityTokenReference>
4784           <wse:SecurityTokenReference wsu:id="MsgBoxEncCert"
4785 wse:Usage="http://www.osci.eu/2008/05/common/names/TokenUsage/TransportEncryption"
4786   osci:Role="http://www.osci.eu/2008/05/common/names/role/MsgBox">
4787             <wse:Reference URI="REPLACE_WITH_ACTUAL_URL_to_MsgBoxEncCert"
4788 ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-
4789 profile-1.0#X509v3"/>
4790           </wse:SecurityTokenReference>

```

```

4791     <wsse: SecurityTokenReference wsu: id="MsgBoxSigCert"
4792     wsse: Usage="http://www.osci.eu/2008/05/common/names/TokenUsage/ReceiptSigning"
4793     osci: Role="http://www.osci.eu/2008/05/common/names/role/MsgBox">
4794         <wsse: Reference URI="REPLACE_WITH_ACTUAL_URL_to_MsgBoxSigCert"
4795         ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-
4796         profile-1.0#X509v3"/>
4797     </wsse: SecurityTokenReference>
4798     </wsp: ALL>
4799     </osci: X509CertificateAssertion>
4800 </wsp: Policy>
4801
4802 <wsp: Policy wsu: id="ServicesAssertion">
4803     <osci: ObsoleteAfterAssertion
4804     PolicyRef="http://www.OSCI.ExmapleEndpoint/MsgReturnPolicy.html">
4805         <osci: MsgReturnDays>7</osci: MsgReturnDays>
4806         </osci: ObsoleteAfterAssertion>
4807     </wsp: Policy>
4808
4809 <wsp: Policy wsu: id="AuthLevelPolicy">
4810     <fi mac: Authentication>urn:de:egov:names:firm1.0:securitylevel:high</fi mac: Authentication>
4811
4812     <fi mac: Registration>urn:de:egov:names:firm1.0:securitylevel:veryhigh</fi mac: Registration>
4813
4814     </wsp: Policy>
4815 </wsp: ALL>
4816 </wsp: Policy>

```

4819 Listing 1: ExampleEndpointOSCIPolicy.xml

4820

## Appendix D. Change History

4821

Version	as of	Author	Changes made in chapter / Comments
2.0.1	04/07/13	A. Wall, J. Apitzsch	Finalization, formal QA
2.0.2	14/10/14 – 13/03/15	J. Apitzsch	<ul style="list-style-type: none"> <li>• Changes to 2.9.1 according introduction section</li> <li>• Changes to various chapters according disposition of comments(commenting round 1)</li> <li>• Disposition of comments (round 2):               <ul style="list-style-type: none"> <li>- oscimeta:MessageMetaData – codelist adoption revised again according alignment with XTA project experts</li> <li>- SAML1 usage optional now, as outphasing version</li> <li>- In chapter 7, now referencing to relevant BSI Technical Guidelines, as suggested by comments of BKA</li> </ul> </li> <li>• Mandatory attribute @payloadSchema added to element &lt;MessageType&gt; in oscimeta:MessageMetaData</li> <li>• &lt;ServiceQuality&gt; in oscimeta:MessageMetaData defined as non-empty string now</li> </ul>

4822

## 4823 **Appendix E. Acknowledgements**

4824 Revision 2.01 of the OSCI 2.0 Transport specification first of all profited from work done in the years  
4825 2012-2013 by the technical working group of the project "XTA" of the German "IT-Planungsrat".

4826 For details see: [http://www.it-  
4827 planungsrat.de/SharedDocs/Downloads/DE/Entscheidungen/7.\\_Sitzung/TOP\\_23\\_Anlage%202\\_XTA\\_  
4828 Projektauftrag\\_Langfassung.pdf?\\_\\_blob=publicationFile](http://www.it-planungsrat.de/SharedDocs/Downloads/DE/Entscheidungen/7._Sitzung/TOP_23_Anlage%202_XTA_Projektauftrag_Langfassung.pdf?__blob=publicationFile).

4829 For revision 2.0.2, thanks have to be given for critical review and fruitful suggestions by members of  
4830 the XTA project again, as well as experts from Bayerisches Staatsministerium der Finanzen für  
4831 Landesentwicklung und Heimat, BKA - ITD-S-Nationale-Gremien, BSI, Bundeszentralamt für Steuern,  
4832 cit GmbH, Dataport AöR, Governikus KG, init AG, Sächsisches Staatsministerium der Justiz und für  
4833 Europa, Senator für Inneres und Sport, Bremen.