OSCI-Transport, Version 2.0.1

– Web Services Profiling and Extensions Specification –

Koordinierungsstelle für IT-Standards (KoSIT)

Coordination Office for IT-Standards

Version 2.0.1

Last edited July 4, 2013
This is the approved final version of the OSCI Transport Version 2.0.1 Web Services Profiling and Extensions Specification. Minor clarifications may be eligible, which could result from perceptions made in the implementation and/or rollout process. These will be published in future editions of this document. An overview of changes made to initial the version 2.0.1 of this specification is provided in Appendix D.

The latest edition will always be available at:


Editor of this document:

Jörg Apitzsch, bremen online services GmbH & Co. KG (bos), ja@bos-bremen.de

Quality Assurance:

Ralf Lindemann, bremen online services GmbH & Co. KG (bos), rl@bos-bremen.de
Andrew Wall, bremen online services GmbH & Co. KG (bos), awa@bos-bremen.de

Further contributors are listed in Appendix E.

Comments and questions may be addressed to:

Ms Beate Schulte
Die Senatorin für Finanzen
–
02 – Zentrales IT-Management und E-Government

Koordinierungsstelle für IT-Standards
Schillerstr. 1
28195 Bremen, Germany
Tel.: +49 421 361 19739
E-Mail: beate.schulte@finanzen.bremen.de

or to the editor directly.
Table of Contents

35  1 Introduction.................................................................................................................6
36  2 Document Structure .................................................................................................7
37  3 Document Conventions ...........................................................................................8
38  3.1 Notational Conventions .............................................................................................8
39  3.2 XML Namespaces .....................................................................................................10
40  4 Specification Conformance .......................................................................................11
41  4.1 Conformance Requirements .....................................................................................11
42  4.2 OSCI Roles and Conformance Targets .....................................................................11
43  5 SOAP Version, Transport and Fault Binding ...........................................................13
44  5.1 General processing error .........................................................................................13
45  5.2 Fault Delivery, Logging and Escalation ....................................................................14
46  6 Addressing Endpoints ...............................................................................................15
47  6.1 Use of WS-Addressing ............................................................................................15
48  6.1.1 Endpoint Reference .............................................................................................15
49  6.1.2 Addressing Properties – SOAP Binding ..............................................................17
50  6.2 Non addressable Initiators and use of WS MakeConnection ..................................20
51  6.3 Addressing faults ..................................................................................................20
52  7 Message Security, Authentication and Authorization .............................................21
53  7.1 WS Security Header Block .....................................................................................21
54  7.2 XML Digital Signature ............................................................................................21
55  7.2.1 Restrictions to WS-I Basic Security Profiling ...................................................21
56  7.2.2 Format of XML Digital Signatures used for Documents ..................................22
57  7.3 XML Encryption .....................................................................................................25
58  7.3.1 End-to-end Encryption of Content Data .................................................................25
59  7.3.2 Encryption Cyphersuite Restrictions ..................................................................25
60  7.4 Security Token Types ............................................................................................26
61  7.5 Use of WS-Trust and SAML Token .......................................................................27
62  7.5.1 Authentication Strongness ...................................................................................27
63  7.5.2 WS-Trust Messages ...........................................................................................29
64  7.5.3 Issued SAML-Token Details ...............................................................................34
65  7.5.4 Authentication for Foreign Domain Access .......................................................36
66  7.5.5 SAML-Token for Receipt/Notification Delivery ..................................................36
67  8 OSCI Specific Extensions ..........................................................................................40
68  8.1 Message Flow Time Stamping .................................................................................40
69  8.2 Accessing Message Boxes .....................................................................................41
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.2.1</td>
<td>MsgBoxFetchRequest</td>
<td>41</td>
</tr>
<tr>
<td>8.2.2</td>
<td>MsgBoxStatusListRequest</td>
<td>44</td>
</tr>
<tr>
<td>8.2.3</td>
<td>MsgBoxResponse</td>
<td>46</td>
</tr>
<tr>
<td>8.2.4</td>
<td>MsgBoxGetNextRequest</td>
<td>50</td>
</tr>
<tr>
<td>8.2.5</td>
<td>MsgBoxCloseRequest</td>
<td>52</td>
</tr>
<tr>
<td>8.2.6</td>
<td>Processing Rules for MsgBoxGetNext/CloseRequest</td>
<td>53</td>
</tr>
<tr>
<td>8.3</td>
<td>Receipts</td>
<td>53</td>
</tr>
<tr>
<td>8.3.1</td>
<td>Demanding Receipts</td>
<td>54</td>
</tr>
<tr>
<td>8.3.2</td>
<td>Receipt Format and Processing</td>
<td>57</td>
</tr>
<tr>
<td>8.3.3</td>
<td>Submission and Relay Receipt</td>
<td>62</td>
</tr>
<tr>
<td>8.3.4</td>
<td>Fetched Notification</td>
<td>62</td>
</tr>
<tr>
<td>8.3.5</td>
<td>Additional Receipt/Notification Demand Fault Processing Rules</td>
<td>64</td>
</tr>
<tr>
<td>8.4</td>
<td>Message Meta Data</td>
<td>65</td>
</tr>
<tr>
<td>8.4.1</td>
<td>Re-used Type Definitions</td>
<td>65</td>
</tr>
<tr>
<td>8.4.2</td>
<td>Description of Message Meta Data Header</td>
<td>69</td>
</tr>
<tr>
<td>8.5</td>
<td>X.509-Token Validation on the Message Route</td>
<td>76</td>
</tr>
<tr>
<td>8.5.1</td>
<td>X.509-Token Container</td>
<td>76</td>
</tr>
<tr>
<td>8.5.2</td>
<td>X.509-Token Validation Results</td>
<td>79</td>
</tr>
<tr>
<td>8.5.3</td>
<td>Verification of XKMS Validate Result Signatures</td>
<td>79</td>
</tr>
<tr>
<td>8.6</td>
<td>General Processing of Custom Header Faults</td>
<td>80</td>
</tr>
<tr>
<td>9</td>
<td>Constituents of OSCI Message Types</td>
<td>81</td>
</tr>
<tr>
<td>9.1</td>
<td>osci:Request</td>
<td>82</td>
</tr>
<tr>
<td>9.2</td>
<td>osci:Response</td>
<td>84</td>
</tr>
<tr>
<td>9.3</td>
<td>MsgBoxFetchRequest</td>
<td>86</td>
</tr>
<tr>
<td>9.4</td>
<td>MsgBoxStatusListRequest</td>
<td>87</td>
</tr>
<tr>
<td>9.5</td>
<td>MsgBoxResponse</td>
<td>88</td>
</tr>
<tr>
<td>9.6</td>
<td>MsgBoxGetNextRequest</td>
<td>89</td>
</tr>
<tr>
<td>9.7</td>
<td>MsgBoxCloseRequest</td>
<td>91</td>
</tr>
<tr>
<td>10</td>
<td>Policies and Metadata of Communication Nodes and Endpoints</td>
<td>93</td>
</tr>
<tr>
<td>10.1</td>
<td>General Usage of Web Service Description Language</td>
<td>93</td>
</tr>
<tr>
<td>10.1.1</td>
<td>WSDL and Policies for MEP Synchronous Point-To-Point</td>
<td>93</td>
</tr>
<tr>
<td>10.1.2</td>
<td>WSDL and Policies for Asynchronous MEPs via Message Boxes</td>
<td>94</td>
</tr>
<tr>
<td>10.2</td>
<td>OSCI Specific Characteristics of Endpoints</td>
<td>94</td>
</tr>
<tr>
<td>10.2.1</td>
<td>Certificates used for Signatures and Encryption</td>
<td>94</td>
</tr>
<tr>
<td>10.2.2</td>
<td>Endpoint Services and Limitations</td>
<td>98</td>
</tr>
<tr>
<td>10.3</td>
<td>WS Addressing Metadata and WS MakeConnection</td>
<td>100</td>
</tr>
<tr>
<td>10.4</td>
<td>WS Reliable Messaging Policy Assertions</td>
<td>101</td>
</tr>
</tbody>
</table>

© 2013 Koordinierungsstelle für IT-Standards / Coordination Office for IT Standards
107  10.5  MTOM Policy Assertion..............................................................101
108  10.6  WS Security Profile and Policy Assertions ..............................................................102
109  10.6.1  Endpoint Policy Subject Assertions ..............................................................102
110  10.6.2  Message Policy Subject Assertions ..............................................................103
111  10.6.3  Algorithm Suite Assertions ..............................................................104
112  11  Applying End-to-end Encryption and Digital Signatures on Content Data .....................105
113  12  Indices .............................................................................................................106
114  12.1  Tables .............................................................................................................106
115  12.2  Pictures .............................................................................................................106
116  12.3  OSCI specific faults .........................................................................................106
117  12.4  Listings .............................................................................................................107
118  13  References .........................................................................................................108
119  13.1  Normative .........................................................................................................108
120  13.2  Informative .......................................................................................................111
121  Appendix A. Schema OSCI Transport 2.01 ........................................................................112
122  Appendix B. OSCI Transport 2.01 – Schema MessageMetaData ........................................118
123  Appendix C. Example: OSCI Endpoint Metadata Instance .............................................124
124  Appendix D. Example Signature Element .......................................................................126
125  Appendix E. Change History .......................................................................................128
126  Appendix F. Acknowledgements ...................................................................................129
1 Introduction

This version 2.0.1 of the Web Services Profiling and Extensions Specification Online Service Computer Interface Transport (OSCI) replaces the former version 2.0, which was made up of four documents:

1. "OSCI-Transport 2.0 – Functional Requirements and Design Objectives"
2. "OSCI-Transport 2 – Features and Architecture Overview"
3. "OSCI Transport 2.0 – Web Services Profiling and Extensions Specification"
4. "OSCI Transport 2 – Glossary"

These four documents are accomplished by a common comprehensive glossary:

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

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

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

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

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

Due to the proliferation of differing platforms and technologies in the e-government, it is essential to ensure that the different web service implementations are interoperable, regardless of the underlying implementation and operation technology. Therefore, we mainly rely on the work, which is done by the Web Services Interoperability Organization⁷, where a profiling is compiled of the major web services specifications under the aspect of best practices for web services interoperability. If needed to satisfy the underlying OSCI requirements, we define further restrictions and processing rules in addition to this profiling. This is outlined in the chapters [5 through 7].

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

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

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

⁷ see http://www.ws-i.org/
3 Document Conventions

3.1 Notational Conventions

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

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

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

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

- An element extensibility point is referred to using {any} in place of the element name. This indicates that any element name can be used, from any namespace other than the osci: or osci21: namespaces.
- An attribute extensibility point is referred to using @{any} in place of the attribute name. This indicates that any attribute name from any namespace can be used.

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

Rnnnn - Statement text here

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

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

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

- Mandatory constituents have continuous lines, optional ones are marked dashed.
- If present, arrows on the left diagram side mark transport encryption requirements, those on the right transport signature requirements. If not present, general according requirements are described in textual form.
- Encrypted message parts are marked by a hatched background.
- All arrows illustrate transport encryption and signature based on WS Security asymmetrical binding; if symmetrical binding is used, encryption and signature is applied using HTTPS.
For an explanation of used abbreviations and terms see the additional document "OSCI Transport 2.0 – Glossary".

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

The following XML namespaces are referenced:

<table>
<thead>
<tr>
<th>Prefix</th>
<th>XML Namespace</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>ds</td>
<td><a href="http://www.w3.org/2000/09/xmldsig#">http://www.w3.org/2000/09/xmldsig#</a></td>
<td>[XMLDSIG]</td>
</tr>
<tr>
<td>dss</td>
<td>urn:oasis:names:tc:dss:1.0:core:schema</td>
<td>[DSS]</td>
</tr>
<tr>
<td>fimac</td>
<td>urn:de:egov:names:fim:1.0:authenticationcontext²</td>
<td>[SAFE]</td>
</tr>
<tr>
<td>osci</td>
<td><a href="http://www.osci.eu/ws/2008/05/transport">http://www.osci.eu/ws/2008/05/transport</a></td>
<td>This document</td>
</tr>
<tr>
<td>osci21</td>
<td><a href="http://www.osci.eu/ws/2013/02/transport">http://www.osci.eu/ws/2013/02/transport</a></td>
<td>This document</td>
</tr>
<tr>
<td>s12</td>
<td><a href="http://www.w3.org/2003/05/soap-envelope">http://www.w3.org/2003/05/soap-envelope</a></td>
<td>[SOAP12]</td>
</tr>
<tr>
<td>samlac</td>
<td>urn:oasis:names:tc:SAML:2.0:ac</td>
<td>[SAMLAC]</td>
</tr>
<tr>
<td>saml1</td>
<td>urn:oasis:names:tc:SAML:1.0:assertion</td>
<td>[SAML1]</td>
</tr>
<tr>
<td>saml2</td>
<td>urn:oasis:names:tc:SAML:2.0:assertion</td>
<td>[SAML2]</td>
</tr>
<tr>
<td>wsa</td>
<td><a href="http://www.w3.org/2005/08/addressing">http://www.w3.org/2005/08/addressing</a></td>
<td>[WSA]</td>
</tr>
<tr>
<td>wsaw</td>
<td><a href="http://www.w3.org/2006/05/addressing/wsd1">http://www.w3.org/2006/05/addressing/wsd1</a></td>
<td>[WSAW]</td>
</tr>
<tr>
<td>wsdl</td>
<td><a href="http://www.w3.org/ns/wsd1-instance">http://www.w3.org/ns/wsd1-instance</a></td>
<td>[WSDL]</td>
</tr>
<tr>
<td>wsdl11</td>
<td><a href="http://schemas.xmlsoap.org/wsd1/">http://schemas.xmlsoap.org/wsd1/</a></td>
<td>[WSDL11]</td>
</tr>
<tr>
<td>wsmc</td>
<td><a href="http://docs.oasis-open.org/ws-rx/wsmc/200702">http://docs.oasis-open.org/ws-rx/wsmc/200702</a></td>
<td>[WSMC]</td>
</tr>
<tr>
<td>wsp</td>
<td><a href="http://www.w3.org/ns/ws-security">http://www.w3.org/ns/ws-security</a></td>
<td>[WSPF], [WSPA]</td>
</tr>
<tr>
<td>wspmtom</td>
<td><a href="http://docs.oasis-open.org/ws-rx/wsrmp/200702">http://docs.oasis-open.org/ws-rx/wsrmp/200702</a></td>
<td>[MTOMP]</td>
</tr>
<tr>
<td>wsrm</td>
<td><a href="http://docs.oasis-open.org/ws-rx/wsrmp/200702">http://docs.oasis-open.org/ws-rx/wsrmp/200702</a></td>
<td>[WSRM]</td>
</tr>
<tr>
<td>wss</td>
<td><a href="http://www.w3.org/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">http://www.w3.org/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd</a></td>
<td>[WSS]</td>
</tr>
<tr>
<td>wssp</td>
<td><a href="http://www.w3.org/2004/01/oasis-200401-wss-wssecurity-policy/200702">http://www.w3.org/2004/01/oasis-200401-wss-wssecurity-policy/200702</a></td>
<td>[WSSP]</td>
</tr>
<tr>
<td>wsu</td>
<td><a href="http://www.w3.org/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">http://www.w3.org/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd</a></td>
<td>[WSS]</td>
</tr>
<tr>
<td>wst</td>
<td><a href="http://docs.oasis-open.org/ws-rx/wst-utility/200512">http://docs.oasis-open.org/ws-rx/wst-utility/200512</a></td>
<td>[WST]</td>
</tr>
<tr>
<td>xenc</td>
<td><a href="http://www.w3.org/2001/04/xmlenc#">http://www.w3.org/2001/04/xmlenc#</a></td>
<td>[XENC]</td>
</tr>
<tr>
<td>xkms</td>
<td><a href="http://www.w3.org/2002/03/xkms#">http://www.w3.org/2002/03/xkms#</a></td>
<td>[XKMS]</td>
</tr>
<tr>
<td>xkmsEU</td>
<td><a href="http://uri.peppol.eu/xkmsExt/v2#">http://uri.peppol.eu/xkmsExt/v2#</a></td>
<td>[XKMSEU]</td>
</tr>
<tr>
<td>xs</td>
<td><a href="http://www.w3.org/2001/XMLSchema">http://www.w3.org/2001/XMLSchema</a></td>
<td>[XMLSchema]</td>
</tr>
</tbody>
</table>

² Preliminary namespace for a SAML AuthnContext extension; proposal subject to standardization in Germany

© 2013 Koordinierungsstelle für IT-Standards / Coordination Office for IT Standards
4 Specification Conformance

4.1 Conformance Requirements

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

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

- http://www.osci.eu/ws/2008/05/osci-transport
- http://www.osci.eu/ws/2013/02/osci-transport
- http://www.w3.org/2002/03/xkms#
- http://uri.peppol.eu/xkmsExt/v2#.

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

4.2 OSCI Roles and Conformance Targets

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

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

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

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

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

- osci:Request, osci:Response, MsgBoxFetchRequest, MsgBoxResponse, MsgBoxStatusListRequest, MsgBoxGetNextRequest, MsgBoxCloseRequest

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

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

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

**AUTHOR** (synonym: Requester, Source Application) – end point instance that wishes to use a PROVIDER entity (OSCI-term: READER) web service, providing according message PAYLOAD and initial message transport attributes. An author uses an INITIATOR instance for dispatching messages.
READER (synonym: (Service) Provider, Target Application) – end point instance providing services for AUTHORS, acting on the request message PAYLOAD and expected to produce related response messages. A READER uses a RECIPIENT instance for receiving messages.

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

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

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

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

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

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

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

SAML-TOKEN – Security Token as defined by SAML.

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

Figure 1: Actors and nodes involved in the message flow

Olive marked nodes and message routes were addressed by version 2.0 of OSCI Transport, in version 2.01 now extended by nodes marked turquoise. The message routes named OSCI do not need the full set of WS-* SOAP headers, these are mostly based on the header osci21:MessageMetaData described in chapter [8.4].

Note: For the realisation of the OSCI route, implementations may choose a network connection based on SOAP, a local API interface, or e.g. products offering SOA bus means.
5  SOAP Version, Transport and Fault Binding

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

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

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

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

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

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

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

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

<table>
<thead>
<tr>
<th>Sub Element</th>
<th>Property Label</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>/Fault/Code/Value</td>
<td>[Code]</td>
<td>Sender</td>
</tr>
<tr>
<td>/Fault/Code/Value/Subcode</td>
<td>[Subcode]</td>
<td>A local QName assigned to the fault</td>
</tr>
<tr>
<td>/Fault/Reason/Text</td>
<td>[Reason]</td>
<td>The English language reason explanation</td>
</tr>
</tbody>
</table>

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

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

### 5.1 General processing error

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

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

<table>
<thead>
<tr>
<th>Fault 1: ProcessingException</th>
<th>[Code] Receiver</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[Subcode] ProcessingException</td>
</tr>
<tr>
<td></td>
<td>[Reason] Unspecific processing error</td>
</tr>
</tbody>
</table>
Implementations MAY provide second-level details fields, e.g. a stack trace, if this information does not lead to security vulnerabilities (see advise above).

5.2 Fault Delivery, Logging and Escalation

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

Though, situations exist where the possibility to deliver this information to the initiating node of the underlying message does not exist. In this case, appropriate escalation mechanisms MUST be foreseen by conformant implementations to signal such situations to the system monitoring environment / operating personal; follow-up of this situation is up to the operating policies³.

---

³ Those should be made available online for all possible communication partners. Details are not addressed by this document.
6 Addressing Endpoints

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

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

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

6.1 Use of WS-Addressing

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

6.1.1 Endpoint Reference

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

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

<wsa:EndpointReference>
  <wsa:Address> xs:anyURI </wsa:Address>
  <wsa:ReferenceParameters>
    <osci:TypeOfBusinessScenario>
      <osci:TypeOfBusinessScenarioType>
        <xs:simpleContent>
          <xs:extension base="xs:anyURI">
            <xs:attribute ref="wsa:IsReferenceParameter" use="optional"/>
          </xs:extension>
        </xs:simpleContent>
      </osci:TypeOfBusinessScenarioType>
    </osci:TypeOfBusinessScenario>
  </wsa:ReferenceParameters>
  <wsa:Metadata> ( xmlns:wsdli="http://www.w3.org/ns/wsdl-instance"
    wsdli:wsdlLocation= "xs:anyURI xs:anyURI"> ) | xs:any *
  </wsa:Metadata> ?
</wsa:EndpointReference>

<documentation>

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

© 2013 Koordinierungsstelle für IT-Standards / Coordination Office for IT Standards
Any endpoint SHOULD expose the types of business scenarios which it actually is able to serve in WSDL [WSDL11] format. An XML schema definition for the Content Data to be carried in the SOAP body of the message MUST be bound to the concrete tagged type of business scenario. Following the WSDL binding of WS-Addressing [WSAW], each `osci:TypeOfBusinessScenario` corresponds to a specific port [WSDL11] respective endpoint [WSDL20].

`/osci:TypeOfBusinessScenario`

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

`/osci:TypeOfBusinessScenario/@wsa:IsReferenceParameter`

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

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

<table>
<thead>
<tr>
<th>Type of business scenario URI</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.osci.eu/ws/2008/05/common/urn/messageTypes/Receipt">http://www.osci.eu/ws/2008/05/common/urn/messageTypes/Receipt</a></td>
<td>Receipt type messages</td>
</tr>
<tr>
<td><a href="http://www.osci.eu/ws/2008/05/common/urn/messageTypes/Notification">http://www.osci.eu/ws/2008/05/common/urn/messageTypes/Notification</a></td>
<td>Notification type messages</td>
</tr>
<tr>
<td><a href="http://www.osci.eu/ws/2008/05/common/urn/messageTypes/Fault">http://www.osci.eu/ws/2008/05/common/urn/messageTypes/Fault</a></td>
<td>Fault type messages</td>
</tr>
</tbody>
</table>

Table 2: Predefined business scenario types

The following type of business scenario SHOULD be served by OSCI endpoints, which are intended to be able to support a common mail-style data exchange, optional carrying any type of attachments:

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

`/wsa:MetaData`

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

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

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

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

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

---

4 The according content data schemes are published together with its specification at http://www1.osci.de/sixcms/detail.php?gsid=bremen76.c.2422.de

5 Details are defined in chapter [10.2.1]
Availability of qualified time stamping service and policies, those apply here (as exposed in `/osci:QualTSPAssertion`).

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

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

### 6.1.2 Addressing Properties – SOAP Binding

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

```xml
<wsa:To> xs:anyURI </wsa:To>
<wsa:From> wsa:EndpointReferenceType </wsa:From> ?
<wsa:ReplyTo> wsa:EndpointReferenceType </wsa:ReplyTo> ?
<wsa:FaultTo> wsa:EndpointReferenceType </wsa:FaultTo> ?
<wsa:MessageID> xs:anyURI </wsa:MessageID>
<wsa:RelatesTo RelationshipType="xs:anyURI">xs:anyURI</wsa:RelatesTo> *
<wsa:ReferenceParameters>xs:any*/</wsa:ReferenceParameters>
```

Description of outline above:

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

`/wsa:From`  
As OSCI is designed for authoritative communication, an OSCI message SHOULD carry at most one SOAP header element `wsa:From` of type `wsa:EndpointReferenceType`.

If carried, the issuer of this message MUST expose here the EPR where he is able to accept `osci:Request` messages according to R0120, R0130; it SHOULD carry the same entries as `/wsa:ReplyTo`.

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

---

6 See chapter [10.2.2]
R0140: in case of non-anonymous and/or asynchronous scenarios messages of type osci:Request MUST carry exactly one SOAP header element wsa:ReplyTo of type wsa:EndpointReferenceType. This MUST contain the concrete EPR of the endpoint according to R0120, R0130; it denotes the final destination to which the Recipient MUST deliver the response. The wsa:ReferenceParameters of this EPR SHOULD be the same as bound to the address element wsa:To.

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

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

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

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

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

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

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

### wsa:Action URIs assigned to OSCI Message Types

<table>
<thead>
<tr>
<th>URI</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.osci.eu/ws/2008/05/transport/urn/messageTypes/osci:Request">http://www.osci.eu/ws/2008/05/transport/urn/messageTypes/osci:Request</a></td>
</tr>
<tr>
<td><a href="http://www.osci.eu/ws/2008/05/transport/urn/messageTypes/osci:Response">http://www.osci.eu/ws/2008/05/transport/urn/messageTypes/osci:Response</a></td>
</tr>
<tr>
<td><a href="http://www.osci.eu/ws/2008/05/transport/urn/messageTypes/MsgBoxFetchRequest">http://www.osci.eu/ws/2008/05/transport/urn/messageTypes/MsgBoxFetchRequest</a></td>
</tr>
<tr>
<td><a href="http://www.osci.eu/ws/2008/05/transport/urn/messageTypes/MsgBoxStatusListRequest">http://www.osci.eu/ws/2008/05/transport/urn/messageTypes/MsgBoxStatusListRequest</a></td>
</tr>
<tr>
<td><a href="http://www.osci.eu/ws/2008/05/transport/urn/messageTypes/MsgBoxResponse">http://www.osci.eu/ws/2008/05/transport/urn/messageTypes/MsgBoxResponse</a></td>
</tr>
</tbody>
</table>
Table 3: Predefined URIs for the WS Addressing Action element

<table>
<thead>
<tr>
<th>URI</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.osci.eu/ws/2008/05/transport/urn/messageTypes/MsgBoxGetNextRequest">http://www.osci.eu/ws/2008/05/transport/urn/messageTypes/MsgBoxGetNextRequest</a></td>
</tr>
<tr>
<td><a href="http://www.osci.eu/ws/2008/05/transport/urn/messageTypes/MsgBoxCloseRequest">http://www.osci.eu/ws/2008/05/transport/urn/messageTypes/MsgBoxCloseRequest</a></td>
</tr>
</tbody>
</table>

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

Fault 2: AddrWrongActionURI

<table>
<thead>
<tr>
<th>[Code]</th>
<th>Sender</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Subcode] AddrWrongActionURI</td>
<td></td>
</tr>
<tr>
<td>[Reason]</td>
<td>Invalid Action header URI value</td>
</tr>
</tbody>
</table>

/wsa:MessageID

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

/wsa:RelatesTo *

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

/wsa:RelatesTo/@RelationshipType ?

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

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

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

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

Fault 3: AddrWrongTypeOfBusinessScenario

<table>
<thead>
<tr>
<th>[Code]</th>
<th>Sender</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Subcode] AddrWrongTypeOfBusinessScenario</td>
<td></td>
</tr>
<tr>
<td>[Reason]</td>
<td>Type of Business Scenario missing or not accepted</td>
</tr>
</tbody>
</table>
6.2 Non addressable Initiators and use of WS MakeConnection

Non-addressable initiators themselves can create outbound connections but cannot accept connections from systems outside their network. This may be for reasons of network topology (i.e. NATs), security (i.e. firewalls), or the like. In the view of the OSCI topology, such initiators even have no MsgBox service available where asynchronous response messages can be targeted to. SOAP supports non-addressable clients by leveraging HTTP to take advantage of this fact. Non-addressable SOAP clients create an outbound connection to a server, send the request message over this connection, then read the corresponding response from that same connection (this response channel is referred to as "the HTTP back-channel"). This is why non-addressable clients operate synchronously. The response can be delivered in the HTTP backchannel of the request. For this behaviour, WS-Addressing specifies the anonymous URI to be carried in the /ReplyTo EPR: "http://www.w3.org/2005/08/addressing/anonymous".

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

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

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

6.3 Addressing faults

The WS Addressing fault handling defined in [WSASOAP], chapter 6 "Faults" applies. For general fault handling, see chapter [5.2].
7 Message Security, Authentication and Authorization

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

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

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

- Possible use of transport layer mechanisms (HTTP over SSL/TLS); if useable the profiling of [WSI-BSP11], chapter 3 “Transport Layer Mechanisms” MUST be applied.¹

- If message layer mechanisms must be used: Which message parts have to be encrypted and signed as well as security token to be used for these purposes?

- What kind of token for authentication and authorization must be provided in a message?

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

7.1 WS Security Header Block

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

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

- Supported security token types, outlined in chapter [7.4].

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

\[ R0200 \] - A SOAP header /wss:Security MUST contain exactly one element /wsu:Timestamp. This supersedes R3227 of [WSI-BSP11].²

7.2 XML Digital Signature

7.2.1 Restrictions to WS-I Basic Security Profiling

The profiling of [WSI-BSP11], chapter 9 “XML-Signature” MUST be applied with the following restrictions going beyond them:³

\[ R0300 \] - Transform algorithm "http://www.w3.org/2001/10/xml-exc-c14n#" is RECOMMENDED. This supersedes R5423 and R5412 of [WSI-BSP11] to clarify; this is

¹ Applicable TLS/SSL versions and cyphersuites are defined here
² “MUST NOT contain more than one” is profiled by [WSI-BSP11]
³ Recommendation: These restrictions SHOULD be regarded by SAML-Token issuers, too.
As the digest algorithm SHA-1 is seen to be weak meanwhile, one of following digest method algorithms MUST be used:

**Digest method algorithms**

- [http://www.w3.org/2001/04/xmlenc#sha256](http://www.w3.org/2001/04/xmlenc#sha256)
- [http://www.w3.org/2001/04/xmlenc#sha512](http://www.w3.org/2001/04/xmlenc#sha512)

<table>
<thead>
<tr>
<th>Table 4: Digest method: allowed algorithm identifiers</th>
</tr>
</thead>
<tbody>
<tr>
<td>The use of SHA-256 (<a href="http://www.w3.org/2001/04/xmlenc#sha256">http://www.w3.org/2001/04/xmlenc#sha256</a>) as digest method algorithm is RECOMMENDED. This supersedes R5420 of [WSI-BSP11][10].</td>
</tr>
</tbody>
</table>

For asymmetric signature methods, the following algorithms MUST be used:

**Asymmetric signature method algorithms**

- [http://www.w3.org/2001/04/xmldsig-more#rsa-sha256](http://www.w3.org/2001/04/xmldsig-more#rsa-sha256)
- [http://www.w3.org/2001/04/xmldsig-more#rsa-sha512](http://www.w3.org/2001/04/xmldsig-more#rsa-sha512)

<table>
<thead>
<tr>
<th>Table 5: Signature method: allowed algorithm identifiers</th>
</tr>
</thead>
<tbody>
<tr>
<td>This supersedes R5421 of [WSI-BSP11][11].</td>
</tr>
</tbody>
</table>

**Note:** To ensure downwards compatibility, the verification of signatures produced with [http://www.w3.org/2001/04/xmldsig-more#rsa-ripemd160](http://www.w3.org/2001/04/xmldsig-more#rsa-ripemd160) MUST be supported.

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

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

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

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

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

**R0400** - The detached XML Signature format MUST be used and the signed content, if part of the message (child of SOAP envelope), be referenced by the local (fragment) URI mechanism as defined in [RFC2396]. Referenceable fragments of message parts MUST carry an

---


attribute of type xs:ID. The constraints of the XML 1.0 [XML 1.0] ID type MUST be met.
The generation of unique ID attribute value SHOULD follow [RFC4122], this value
SHOULD be concatenated to a preceding string "uuid:.".\(^{12}\)

A ds:Signature element MUST contain at least one ds:Object child element to carry
the signing time and a reference to the certificate used for signing. The format of this child
element MUST conform to definitions given by [XAdES] and the following restrictions must
apply here:

It MUST contain exactly one child element xades:QualifyingProperties including
the mandatory child element,
xades:SignedProperties/xades:SignedSignatureProperties and an optional
child element xades:UnsignedProperties, which is foreseen to carry a qualified
timestamp over the signature itself in the child element
xades:UnsignedSignatureProperties/xades:SignatureTimeStamp.

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

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

Exclusive canonicalization MUST be used to address requirements resulting from
scenarios where subdocuments are moved between contexts. The URI-Value of the
attribute ds:CanonicalizationMethod/@Algorithm is fixed to
"http://www.w3.org/2001/10/xml-exc-c14n#".\(^{13}\)

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

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

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

```xml
<ds:Signature Id="xs:ID">
  <ds:SignedInfo Id="xs:ID"/>
  <ds:CanonicalizationMethod
    Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
</ds:CanonicalizationMethod>
  <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/>
  <ds:Reference Id="xs:ID"/>
</ds:Signature>
```

---

\(^{12}\) WS-Frameworks may foresee other ID attribute value generation mechanisms

\(^{13}\) See also: R5404 of [WSI-BSP11]
Type="http://uri.etsi.org/011903/v1.1.1/#SignedProperties"
<ds:Transforms/>
<ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256" />
<ds:DigestMethod>
<ds:DigestValue> xs:base64Binary </DigestValue>
<ds:Reference>
(<ds:Reference Id="xs:ID" ?
 Type="xs:anyURI"
 URI="xs:anyURI">
<ds:Transforms/>
<ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc-more#rsa-sha256" />
<ds:DigestMethod>
<ds:DigestValue> xs:base64Binary </DigestValue>
</Reference> ) +
</ds:Reference>
</ds:DigestMethod>
<ds:DigestValue> xs:base64Binary </DigestValue>
<ds:Reference>
</Reference>
</ds:SignedInfo>
<ds:SignatureValue Id="xs:ID" ?> xs:base64Binary </ds:SignatureValue>
<ds:KeyInfo Id="xs:ID" ?>
<ds:RetrievalMethod
 Type="http://www.w3.org/2000/09/xmldsig/X509Data"/>
<ds:X509Data>
<ds:X509IssuerSerial>
<ds:X509IssuerName> xs:string </ds:X509IssuerName>
<ds:X509SerialNumber> xs:integer </ds:X509SerialNumber>
</ds:X509IssuerSerial>
<ds:X509CRL/> ?
</ds:X509Data>
</ds:KeyInfo>
<ds:Object Id="xs:ID">
<xades:QualifyingProperties Target="...">
<xades:SignedProperties>
<xades:SignedSignatureProperties Id="xs:ID">
<xades:SigningTime> xs:dateTime </xades:SigningTime>
<xades:SigningCertificate>
<xades:Cert>
<xades:CertDigest>
<ds:DigestMethod> Algorithm="http://www.w3.org/2001/04/xmlsig-more#rsa-sha256" />
<ds:DigestMethod>
<ds:DigestValue> xs:base64Binary </DigestValue>
</xades:Cert>
</xades:SignedProperties>
</xades:QualifyingProperties>
</ds:Object>
The outline above shows mandatory and optional elements and their cardinality restrictions. For a detailed description of elements and attributes in the outline above, see [XMLDSIG] and [XAdES].

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

### 7.3 XML Encryption

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

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

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

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

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

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

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

#### 7.3.2 Encryption Cyphersuite Restrictions

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

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

Table 6: Symmetric encryption algorithms
Encryption of symmetric keys MUST be performed by means of RSAES-OAEP-ENCRIPT [PKCS#1]. The value of xenc:EncryptionMethod MUST be "http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p"

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

### 7.4 Security Token Types

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

<table>
<thead>
<tr>
<th>Security Token Type</th>
<th>Support</th>
<th>Value of wsse:BinarySecurityToken/@ValueType and wsse:SecurityTokenReference/@ValueType</th>
<th>Profiling Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAMLV2.0</td>
<td>MUST</td>
<td><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></td>
<td>[WSSSAML]</td>
</tr>
<tr>
<td>X.509v3-Certificate</td>
<td>MUST</td>
<td><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></td>
<td>[WSSX509]</td>
</tr>
<tr>
<td>Kerberos</td>
<td>MAY</td>
<td><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></td>
<td>[WSSKERB]</td>
</tr>
<tr>
<td>Username</td>
<td>MAY</td>
<td>Defined in WS-Security as wsse:UsernameToken</td>
<td>[WSSUSER]</td>
</tr>
</tbody>
</table>

Table 7: Security token types – support requirements

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

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

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

  - X.509v3-Certificate token issued by CAs MUST be used for authentication and message security for scenarios allowing anonymous access. Validity of used certificates MUST be verifiable by means of OCSP, LDA,P or CRL.

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

<table>
<thead>
<tr>
<th>Fault 4: AuthnCertNotValid</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Code] Sender</td>
</tr>
<tr>
<td>[Subcode] AuthnCertNotValid</td>
</tr>
<tr>
<td>[Reason] Authentication certificate not stated to be valid</td>
</tr>
</tbody>
</table>

More information about the certificate validation results SHOULD be provided in the fault [Details] property in this case. It is strongly RECOMMENDED to log such faults to be able to detect possible security violation attacks.
Token of type Username and Kerberos MAY be used for authentication and securing messages inside closed communication domains, where security and trust is given by means out of band of this specification.

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

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

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

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

To assure interoperability with WS-Trust/SAML infrastructures rolled out, both SAML Version 1.1 and Version 2.0 SHOULD be support by OSCI implementation.

#### 7.5.1 Authentication Strongness

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

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

[SAML1] respective [SAML2] and [SAMLAC] specify an authentication statement 
\[saml<1|2>::AuthnStatement\] to carry such information. Differentiated authentication context details may be included herein. To simplify processing and interoperability, the following ascending levels for strongness of registration and authentication are defined:

- urn:de:egov:names:fim:1.0:securitylevel:normal
- urn:de:egov:names:fim:1.0:securitylevel:high

---

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

15 Preliminary URIs proposed by the SAFE-Project; subject to standardization activities by German administration.
Each level matches operational rules which must be defined, published, and continuously maintained by appropriate institutions, i.e. government agencies concerned with data protection.\textsuperscript{16} [SAFE] defines extensions to the SAML authentication context element to carry the levels of registration and authentication as follows:

\begin{verbatim}
<samlac:Extension>
  <fimac:SecurityLevel>
    <fimac:Authentication>
      urn:de:egov:names:fim:1.0:securitylevel:normal |
    </fimac:Authentication> |
    <fimac:Registration>
      urn:de:egov:names:fim:1.0:securitylevel:normal |
      urn:de:egov:names:fim:1.0:securitylevel:high |
      urn:de:egov:names:fim:1.0:securitylevel:veryhigh |
    </fimac:Registration> |
  </fimac:SecurityLevel> |
</samlac:Extension>
\end{verbatim}

This outline is preliminary to be seen as normative.

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

Optional container carrying the detail elements.

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

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

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

Fault 6: `AuthnSecurityLevelInsufficient`

[Code] Sender

[Subcode] AuthnSecurityLevelInsufficient

[Reason] Insufficient strongness of authentication or registration

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

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

\textsuperscript{16} 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.
7.5.2 WS-Trust Messages

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

- Issue
- Validate
- Cancel.

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

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

7.5.2.1 Request Security Token (RST)

![Diagram of Request Security Token Message]

Figure 2: Request Security Token Message
SOAP header blocks:

/wsse:Security
This header block MUST be present, carrying message protection data and initiator authentication information according to the security policy of the STS the RST message is targeted to.

/wsse:Security/wsu:Timestamp
According to R0200, this header block MUST be present.

/wsse:Security/[Security-Token]
Security tokens MUST be used for signing and encrypting message parts. ds:KeyInfo elements of subsequent ds:Signature or xenc:EncryptedKey elements MAY point to security tokens carried here.

[Table 7] lists the security token types which MUST or MAY be supported.

Security tokens MUST be embedded or referenced. Referenced tokens MUST be dereferencable by the targeted STS.

The requestors security token MUST be used for signing the above marked message parts.

/wsse:Security/ds:Signature
A signature containing ds:Reference elements for all message parts marked above is to be included in the signature.

/wsse:Security/xenc:EncryptedKey
The RST contained in the SOAP body block MUST be encrypted for the targeted STS.

This is a symmetric key, which MUST be encrypted with the public key of the STS X.509v3 encryption certificate. Rules outlined in chapter [7.3] apply. It is assumed that the STS encryptions certificate is made available to all endpoints outside the STS Trust Domain out of band of this specification.

/wsa:*
All WS-Addressing headers MUST (if in continuous blocks) /MAY (if in dashed blocks) be supplied by the requestor.

/wsa:Action
Depending on the type of WS-Trust request, one of the URIs outlined above MUST be supplied. This URI MUST be adequate to the respective body block content.

The SOAP body block MUST conform to the definitions of WS-Trust, whereby the following restrictions and recommendations apply for the WS-Trust Issue request type.
Figure 3: Request Security Token, Body for Issue Request

/wsp:AppliesTo

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

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

/wst:TokenType

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

- urn:oasis:names:tc:SAML:2.0:assertion
- urn:oasis:names:tc:SAML:1.0:assertion

/wst:KeyType

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

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

/wst:KeySize

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

Use and values of elements marked optional are subject to used STS instance specific policies. Recommendations will be given as part of the amendments to be worked out for this specification in 2009 ff.
7.5.2.2 Request Security Token Response (RSTR)

The SOAP header resembles the one of the RST message:

![Figure 4: Request Security Token Response Message]

Differences to the RST message:

/\texttt{wsse:Security/xenc:EncryptedKey}\n
The RSTRC contained in the SOAP body block MUST be encrypted for the token requestor. This is a symmetric key which MUST be encrypted with the public key of the requestors X.509v3 encryption certificate. Rules outlined in chapter [7.3] apply.

/\texttt{wsa:*}\n
All WS-Addressing headers MUST (if in continuous blocks) /MAY (if in dashed blocks) be supplied by the STS.

/\texttt{wsa:Action}\n
Depending on the type of WS-Trust response, one of the URIs outlined above MUST be supplied. This URI MUST be adequate to the respective body block content.

The decrypted SOAP body block MUST conform to the definitions of WS-Trust. No restrictions or profiling apply.
The SOAP body block MUST conform to the definitions of WS-Trust:

![Diagram of SOAP body block]

Figure 5: Request Security Token, Body for Issue Response

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

- `/wsp:AppliesTo` Carries the value assignment for the desired application scope of the requested security token – copied from the according request element.

- `/wst:TokenType` Carries the token type, which MUST be the one of the according request elements.

- `/wst:RequestedSecurityToken` Carries the requested SAML-Token, including a symmetric key encrypted for the endpoint at which the SAML-Token is needed for authentication purposes. Details are explained in chapter [7.5.3].

- `/wst:RequestedProofToken` Carries information enabling the requestor to deduce the symmetric key. In case the key was generated by the STS solely, this is the key itself. In case computed of two entropy values, this is the algorithm and the element

- `/wst:Entropy ?` MUST be present carrying the entropy value used by the STS for key computation.

- `/wst:LifeTime ?` Optional element carrying the validity duration period of this RSTR; SHOULD be recognized by the requestor and processed according to his needs to avoid using security tokens being/getting invalid.
7.5.3 Issued SAML-Token Details

Figure 6: SAML 2.0 Assertion constituents

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

/saml2:Issuer

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

/saml2:Conditions

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

/saml2:Subject

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

/saml2:Subject/saml2:NameID

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

/saml2:Subject/saml2:SubjectConfirmation

17 For brevity, we only illustrate the SAML Version 2.0 Assertion in this document. For the SAML Version 1.1 Assertion layout, see [SAML1].
This container exposes STS information for the SP enabling it to assure that the SR is the one stated in /saml2:NameID and authorized to use this token.

/saml2:Subject/saml2:SubjectConfirmation/@Method

R0750: Attribute outlining the confirmation method; MUST be the "holder of key" confirmation method.

/saml2:Subject/saml2:SubjectConfirmation/saml12:SubjectConfirmationData

R0760: Presence of this element is REQUIRED; it exposes STS information for the SP enabling it to assure that the SR is the one owning the key for this SAML assertion.

/saml2:Subject/saml2:SubjectConfirmation/saml12:SubjectConfirmationData/ds:key

R0770: This element MUST carry the key in a xenc:EncryptedKey element. The key MUST be encrypted for the SP using the public key of its X.509v3 encryption certificate, which for this purpose MUST be made available to the STS.

NOTE on the endpoint encryption certificate, the SAML token is targeted to:

The access to this certificate through the token issuing STS is of band of this specification; this is a matter of Trust Domain policies and an implementation issue which MUST have no effect on interoperability. No standardized mechanisms are foreseen by WS-Trust, to include a certificate in an RST message for the purpose of key encryption for the SP. It is strongly RECOMMENDED, to relate the /wsp:AppliesTo request value (which might be a pattern, too – see RST body description in chapter [7.5.2.1]) to this encryption certificate.

/saml2:AuthnStatement

R0780: Presence of this element is REQUIRED.

It MUST contain an element /saml2:AuthnContext with an attribute @AuthnInstant outlining the time instant the authentication took place.

/saml2:AuthnContext MUST contain an element /saml2:AuthnContextClassRef outlining the authentication method used by the SR.\(^{18}\)

/saml2:AuthnContext MUST further contain an element /saml2:AuthnContextDecl carrying the extensions for authentication strength as defined in chapter [7.5.1].

/saml2:AttributeStatement ?

Usage of attribute statements of type saml2:AttributeType is RECOMMENDED. In many scenarios subject attributes like affiliation to certain groups or roles are used for the assignment's detailed rights, functions and data access. Hence attributes are specific to application scenarios, their names, values, and semantics are subject to the overall design of a domain information model, which is not addressed by this specification.\(^{19}\)

/ds:Signature

The issuing STS has to sign the whole SAML-Token.

---

\(^{18}\) 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

\(^{19}\) Suggestions for use in German e-government, especially e-justice, are made in [SAFE]
If a SAML token does not match one or more of the formal requirements 0730-0780, the token consuming node MUST generate a fault and discard the message.

### Fault 7: AuthnTokenFormalMismatch

<table>
<thead>
<tr>
<th>Code</th>
<th>Sender</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subcode</td>
<td>AuthnTokenFormalMismatch</td>
</tr>
<tr>
<td>Reason</td>
<td>Authentication token present does not match formal requirements.</td>
</tr>
</tbody>
</table>

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

#### 7.5.4 Authentication for Foreign Domain Access

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

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

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

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

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

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

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

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

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

1. It MUST be requested from the initiator's STS.
2. The according RST message MUST contain the \texttt{wsa:MessageID} and the address of the receiving/notifying node (\texttt{wsp:AppliesTo}) as claims.

3. The symmetric key of the issued SAML-Token MUST be encrypted for the endpoint outlined in the element \texttt{.../wsa:ReplyTo} of the receipt/notification demand; the \texttt{wst:RequestedProofToken} in this case MUST be encrypted for the receiving/notifying node (for use in the following step 6)

4. The issuing STS MUST retain this OneTimeToken for later use and mark it as "unused".

5. The RSTR message returned by the STS MUST be included as separate SOAP header block in the request message.

6. The receiving/notifying node has to use the OneTimeToken included in this RSTR as SAML-Token for the message the receipt/notification is delivered with. Transport signature and encryption MUST be generated based on the symmetric key contained in the \texttt{wst:RequestedProofToken}.

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

7. Decryption of the OneTimeToken's symmetric key.

8. Validation of the signature of the OneTimeToken – the symmetric key MUST be the same the receiving/notifying node used for the transport signature.

9. Validation of the signature of the issuing STS and RST-Validate message containing the OneTimeToken to the STS.

10. If at the issuing STS this OneTimeToken is still marked as "unused", the token is valid.

11. If the RSTR signals valid in validate-response: Acceptance of the message containing the receipt/notification.

12. Message accepting node MUST target a RST/cancel message to the STS to invalidate this OneTimeToken; STS SHOULD discard this token.

The following diagrams illustrate the RST and RSTR for the OneTimeToken, for concrete XML Infoset descriptions see WS-Trust and SAML specifications.
Figure 7: RST for OneTimeToken
Figure 8: RSTR for OneTimeToken
8 OSCI Specific Extensions

8.1 Message Flow Time Stamping

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

```
<osci:MsgTimeStamps wsu:Id="..." ? >
  <osci:ObsoleteAfter> xs:date </osci:ObsoleteAfter> ?
  <osci:Delivery> xs:dateTime </osci:Delivery> ?
  <osci:InitialFetch> xs:dateTime </osci:InitialFetch> ?
  <osci:Reception> xs:dateTime </osci:Reception> ?
</osci:MsgTimeStamps>
```

Description of elements and attributes in the schema overview above:

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

/osci:MsgTimeStamps

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

/osci:MsgTimeStamps/@wsu:Id

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

/oci:MsgTimeStamps/osci:ObsoleteAfter ?

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

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

/oci:MsgTimeStamps/osci:Delivery ?

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

/oci:MsgTimeStamps/osci:InitialFetch ?

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

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

/oci:MsgTimeStamps/osci:Reception ?

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

It MUST be set by a MsgBox node to its actual server time when the Recipient commits the reception of a message through a MsgBoxGetNextRequest or MsgBoxCloseRequest.
8.2 Accessing Message Boxes

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

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

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

Fault 8: MsgSelectorNotSupported

[Code] Sender
[Subcode] MsgSelectorNotSupported
[Reason] Presented selection criteria not supported

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

8.2.1 MsgBoxFetchRequest

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

The normative outline for a MsgBoxFetchRequest request is:

```xml
<s12:Envelope ...>
  <s12:Header ...>
    ...
    <wsa:Action>
      http://www.osci.eu/ws/2008/05/transport/urn/messageTypes/MsgBoxFetchRequest
    </wsa:Action>
    <wsa:MessageID>xs:anyURI</wsa:MessageID>
    <wsa:To>xs:anyURI</wsa:To>
    <osci:TypeOfBusinessScenario wsa:IsReferenceParameter="1">
      xs:anyURI
    </osci:TypeOfBusinessScenario>
    ...
  </s12:Header>
  <s12:Body ...
    <osci:MsgBoxFetchRequest MsgPart="("Envelope" | "Body" | "Header")">
      <osci:MsgSelector newEntry="true" | "false">
        <wsa:MessageID>xs:anyURI</wsa:MessageID> *
        <wsa:RelatesTo>xs:anyURI</wsa:RelatesTo> *=
        <osci:MsgBoxEntryTimeFrom>xs:dateTime</osci:MsgBoxEntryTimeFrom> *
        <osci:Extension>xs:anyType</osci:Extension> *
      </osci:MsgSelector> *
    </osci:MsgBoxFetchRequest>
  </s12:Body>
</s12:Envelope>
```

The following describes normative constraints on the outline listed above:
The value indicated herein MUST be used for that URI.

The request MUST carry a unique WS-Addressing MessageID.

The address of the MsgBox (request destination) endpoint.

This value of the instantiation of /wsa:ReferenceParameters MUST be supplied for this message type. The value of /osci:TypeOfBusinessScenario is taken as message selection argument and SHOULD match one of those accepted by this endpoint.

If a MsgBoxFetchRequest contains no other arguments for message selection in the SOAP body element /osci:MsgBoxFetchRequest/osci:MsgSelector, the messages to be selected MUST be those which have not yet been fetched. Those are all messages in the addressed MsgBox which have no SOAP header element or a value of zero in the time instant element /osci:MsgTimeStamps/osci:InitialFetched.

They MUST be delivered one per request-/response in a FIFO-manner to the endpoint denoted by /s12:Envelope/s12:Header/wsa:ReplyTo.

The body of this message contains the actual request in a structure:

Container holding detailed selection arguments in addition to /s12:Envelope/s12:Header/oci:TypeOfBusinessScenario above; this element MAY be empty if no further selection criteria shall be provided.

This optional attribute of type xs:NMTOKEN allows the recipient to denote which part of a message shall be returned in the SOAP body of the subsequent MsgBoxResponse messages:

- "Envelope" – returns the whole s12:Envelope container of the selected messages as child element of the SOAP body block of the response message.
- "Header": whole resulting SOAP header elements are included as child elements of the SOAP body block of the response message.
- "Body": only the original SOAP body child element MUST be included unchanged as child element of the SOAP body of the response message.

Note: This attribute has been introduced with version 2.0.1 of this specification.

If this optional element is present, arguments of the attribute @newEntry and sub-elements MsgSelector MUST be processed as logical AND (after first OR-processing of

---

20 As recommended in chapter [5], a SOAP body is assumed to carry only one child element.
the sequences of MessageIDs in the SOAP body elements ...

.../osci:MessageID and
.../osci:RelatesTo, if present).

/s12:Envelope/s12:Body/osci:MsgBoxFetchRequest/osci:MsgSelector/@newEntry ?

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

/s12:Envelope/s12:Body/osci:MsgBoxFetchRequest/osci:MsgSelector/
osci:MessageID *

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

/s12:Envelope/s12:Body/osci:MsgBoxFetchRequest/osci:MsgSelector/
osci:RelatesTo *

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

osci:MsgBoxEntryTimeFrom ?

If present, this element denotes a value of type xs:date as lower value when a
message has been accepted by a MsgBox service. The resulting search expression is
.../oci:MsgBoxEntryTimeFrom => the value of ...

osci:MsgBoxEntryTimeTo ?

If present, this element denotes a value of xs:date as upper value when a
message has been accepted by a MsgBox service. The resulting search expression is
.../oci:MsgBoxEntryTimeTo <= the value of ...

osci:MsgBoxEntryFrom ?

If present, this element denotes a value of xs:date as lower value when a
message has been accepted by a MsgBox service. The resulting search expression is
.../oci:MsgBoxEntryFrom <= the value of ...

osci:Extension/{any} *

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

Note: For implementations implementing version 2.0.1 and higher of this specification, it is
RECOMMENDED to support XPath-querying based on the header Block
osci21:MessageMetaData introduced with version 2.0.1 of this specification.
Upon receipt and authentication of this message, the MsgBox service MUST locate any message that matches the selection criteria. Only messages originally targeted to this EPR MUST be returned. The search criteria MUST include examinations of the child elements inside the SOAP body element `/osci:MsgSelector`.

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

### 8.2.2 MsgBoxStatusListRequest

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

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

```xml
<s12:Envelope ...>
  <s12:Header ...>
    ... etc...
    <wsa:Action>
      http://www.osci.eu/ws/2008/05/transport/urn/messageTypes/MsgBoxStatusListRequest
    </wsa:Action>
    <wsa:MessageID>xs:anyURI</wsa:MessageID>
    <wsa:To>xs:anyURI</wsa:To>
    <osci:TypeOfBusinessScenario wsa:IsReferenceParameter="1">
      xs:anyURI
    </osci:TypeOfBusinessScenario>
    ... etc...
  </s12:Header>
  <s12:Body ...
    <osci:MsgBoxStatusListRequest maxListItems="xs:positiveInteger">  
      Listform= ("MsgAttributes" | "MessageMetaData")?
      <osci:MsgSelector newEntry="true" | "false">  
        <sci:MessageId> xs:anyURI </sci:MessageId> *
        <sci:RelatesTo> xs:anyURI </sci:RelatesTo> *
        <sci:MsgBoxEntryTimeFrom>
          xs:dateTime
        </sci:MsgBoxEntryTimeFrom> ?
        <sci:MsgBoxEntryTimeTo>
          xs:dateTime
        </sci:MsgBoxEntryTimeTo> ?
        <sci:Extension> xs:anyType </sci:Extension> ?
        <sci:MsgSelector> ?
      </sci:MsgSelector> ?
    </osci:MsgBoxStatusListRequest>
  </s12:Body>
</s12:Envelope>
```

Description of normative constraints on the outline listed above:

/s12:Envelope/s12:Header/wsa:Action

The value indicated herein MUST be used for that URI.

/s12:Envelope/s12:Header/wsa:MessageID

The request MUST carry a unique WS-Addressing MessageID.
The address of the MsgBox (request destination) endpoint.

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

This value of the instantiation of `/wsa:ReferenceParameters` MUST be supplied for this message type. The value of `/osci:TypeOfBusinessScenario` is taken as message selection argument and SHOULD match one of those accepted by this endpoint. As an alternative in this special case a value of "*" MAY be supplied here, to select the message status list for all messages in this(MsgBox instance. Such an entry MUST lead to a message box status list containing all messages, with no regard to a specific addressed business scenario of this endpoint, actually exposed as able to serve.

Only status lists of messages originally targeted to the EPR outlined MUST be returned. If a MsgBoxFetchRequest contains no other arguments for message selection in the SOAP body element `osci:MsgBoxStatusListRequest`, the messages to be selected MUST be those which have not yet been fetched. That are all messages in the addressed MsgBox having no SOAP header element or the value zero in the `.../osci:MsgTimeStamps/osci:InitialFetched` time instant element.

```xml
/s12:Envelope/s12:Body/osci:MsgBoxStatusListRequest
```

Container holding detailed selection arguments in addition to `/s12:Envelope/s12:Header/osci:TypeOfBusinessScenario` above; this element MAY contain no child elements if no further selection criteria shall be provided.

```xml
/s12:Envelope/s12:Body/osci:MsgBoxStatusListRequest/@maxListItems
```

The requestor MAY limit the length of the message status list he expects in the response with this attribute of type `xs:positiveInteger`. A MsgBox service MUST hold the complete list corresponding to the selection criteria and deliver an ID for this list to the requestor together with the response. In subsequent requests (see `MsgBoxGetNextRequest` in chapter [8.2.4]), the requestor is able to request further portions of a selection result with reference to this list.

A MsgBox instance MAY limit the value of `@maxListItems` to any value greater zero.

If provided, a MsgBox instance MUST retain this value – if not decreased by its configured limit - together with the result set until the whole result set is delivered to the requestor or the requestor cancels an iteration sequence (see `MsgBoxCloseRequest` in chapter [8.2.5]).

```xml
/s12:Envelope/s12:Body/osci:MsgBoxStatusListRequest/@ListForm
```

This optional attribute of type `xs:NMTOKEN` with the values listed below allows the recipient to denote whether the `osci:MsgStatusList` to be returned shall contain:

- "MsgAttributes" – for selected messages it returns a sequence of `osci:MsgAttributes` elements as described in [8.2.3.2]. This is the default value and functionality as specified in version 2.0 of this specification
- "MessageMetaData": – for selected messages it returns a sequence of `osci21:MessageMetaData` elements as described in [8.2.3.2].

**Note:** This attribute has been introduced with version 2.0.1 of this specification.

```xml
/s12:Envelope/s12:Body/osci:MsgBoxStatusListRequest/osci:MsgSelector
```
For the content of this complex element, which defines selection criteria for messages, see description in last chapter [\[8.2.1\]].


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

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

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

\[8.2.3\](MsgBoxResponse)

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

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

The normative outline for a MsgBoxResponse header is:

\[<s12:Envelope ...>\]
\[<s12:Header ...>\]
\[...\]
\[<wsa:Action>\]
\[http://www.osci.eu/ws/2008/05/transport/urn:messageTypes/MsgBoxResponse\]
\[</wsa:Action>\]
\[<wsa:MessageID>xs:anyURI</wsa:MessageID>\]
\[<wsa:FaultTo> wsa:EndpointReference </wsa:FaultTo> ?\]
\[...\]
\[<osci:MsgBoxResponse MsgBoxRequestId="xs:anyURI"\]
\[wsu:Id = "xs:ID" ?\]
\[osci:NoMessageAvailable\]
\[reason=\]
\["http://www.osci.eu/ws/2008/05/transport/MsgBox/reasons/NoMatch" |\]
\["http://www.osci.eu/ws/2008/05/transport/MsgBox/reasons/SearchArgsInvalid"|\]
\["http://www.osci.eu/ws/2008/05/transport/MsgBox/reasons/RequestIdInvalid"|\]
\["xs:anyUri" />\]
\[\]
\[<osci:ItemsPending>\]
\[xs:positiveInteger\]
\[</osci:ItemsPending>\]
\[</osci:MsgBoxResponse\]
\[...\]
\[</s12:Header>\]
\[<s12:Body ...>\]
\[<s12:Envelope>\]
Description of normative constraints of the outline listed above (WS-Addressing header elements are to be handled according to chapter [6, Addressing Endpoints]):

/s12:Envelope/s12:Header/wsa:Action

The value indicated herein MUST be used for that URI.

/s12:Envelope/s12:Header/wsa:MessageID

The request MUST carry a unique WS-Addressing MessageID.

/s12:Envelope/s12:Header/wsa:FaultTo

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

/s12:Envelope/s12:Header/osci:MsgBoxResponse

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

The following attributes MUST be set:

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

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

/s12:Envelope/s12:Header/osci:MsgBoxResponse/@wsu:Id

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

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

This element of the choice of .../MsgBoxResponse MUST be set if there are no messages available. Corresponding to the selection criteria there where errors detected in the selection criteria.

The element carries the following attribute:

@reason

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

<table>
<thead>
<tr>
<th>@reason URI</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.osci.eu/ws/2008/05/transport/MsgBox/reasons/NoMatch">http://www.osci.eu/ws/2008/05/transport/MsgBox/reasons/NoMatch</a></td>
<td>No messages matching the search criteria could be found</td>
</tr>
<tr>
<td><a href="http://www.osci.eu/ws/2008/05/transport/MsgBox/reasons/SearchArgsInvalid">http://www.osci.eu/ws/2008/05/transport/MsgBox/reasons/SearchArgsInvalid</a></td>
<td>Error contained in search arguments</td>
</tr>
<tr>
<td><a href="http://www.osci.eu/ws/2008/05/common/urn/MsgBox/reasons/RequestIdInvalid">http://www.osci.eu/ws/2008/05/common/urn/MsgBox/reasons/RequestIdInvalid</a></td>
<td>RequestId of subsequent GetNext- Close- Request is not known or no longer available at the MsgBox instance</td>
</tr>
</tbody>
</table>
The alternative of the choice of …/MsgBoxResponse MUST be set if there are – according to the selection criteria of the request - messages or message status list items pending (not yet deliverable to the requestor in the actual response):

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

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

### 8.2.3.1 MsgBoxResponse - body to MsgBoxFetchRequest

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

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

- A SOAP Envelope with all child elements MUST be build up containing a header block with the ones of the original message, except for header elements, which have initially been targeted to and successfully executed by the MsgBox node, as well as the transport encryption and signature elements, which have been supplied by the Initiator of this message.
- The SOAP header element osci:MsgTimeStamps MUST be inserted (or completed, if present) as described in chapter [8.1].
- All original WS-Addressing, osci:X509TokenContainer, xkms:ValidateResult (inside a xkms:CompoundResult) and original security token and WS-Trust header elements MUST be included.
- If present in the original message, the osci:ReceptionReceiptDemand header element MUST be included.
- If present in the original message, the osci21:MessageMetaData header element MUST be included.
- The original SOAP body child elements MUST be included unchanged as child elements of the SOAP body of this SOAP Envelope to be built up.
- The recipient may have interest in the authentication and authorization data originally carried in the SOAP WS Security header when delivering a message to the recipient's MsgBox. Therefore, this security token MUST be inserted as additional child element into the SOAP header of this SOAP Envelope to be built up.

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

**Rule for OSCI version 2.0.1 conformant implementations:**

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

- "Envelope": rule above applies – whole resulting SOAP envelope MUST be included as child element of the SOAP body block of the response message.
- "Header": whole resulting SOAP header elements MUST be included as child element of the SOAP body block of the response message.
8.2.3.2 MsgBoxResponse - Body to MsgBoxStatusListRequest

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

The normative outline for a MsgBoxStatusList is:

```
<osci:MsgStatusList>
  <osci:MsgAttributes>
    <wsa:MessageID>xs:anyUri</wsa:MessageID>
    <wsa:RelatesTo>xs:anyUri</wsa:RelatesTo> *
    <wsa:From>endpoint-reference</wsa:From> ?
    <osci:TypeOfBusinessScenario>xs:anyUri</osci:TypeOfBusinessScenario>
    <osci:MsgSize>xs:positiveInteger</osci:msgSize>
    <osci:ObsoleteAfterDate>xs:date</osci:ObsoleteAfterDate> ?
    <osci:DeliveryTime>xs:dateTime</osci:DeliveryTime>
    <osci:InitialFetchTime>xs:dateTime</osci:InitialFetchTime> ?
  </osci:MsgAttributes> *
  <osci21:MessageMetaData/> *
</osci:MsgStatusList>
```

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

/osci:MsgStatusList

Container for the items of the list. According to the value of attribute @ListForm of the foregoing osci:MsgBoxStatusListRequest, the list MUST be built of a sequence of ../osci:MsgAttributes (default behaviour) ../osci21:MessageMetaData elements.\(^22\)

For selected messages not carrying a header /osci21:MessageMetaData, .../osci:MsgAttributes MUST be returned.

/oci:MsgStatusList/oci:MsgAttributes *

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

/oci:MsgStatusList/oci:MsgAttributes/wsa:MessageID

MessageID of the message, derived from respective header element.

/oci:MsgStatusList/oci:MsgAttributes/wsa:RelatesTo *

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

/oci:MsgStatusList/oci:MsgAttributes/wsa:From ?

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

---

\(^{21}\) As recommended in chapter [5], a SOAP body is assumed to carry only one child element.

\(^{22}\) Due to downwards compatibility, /oci21:MessageMetaData, .../oci:MsgAttributes are not defined as choice.
This URI denotes the type of addressed business scenario of the intended recipient of the message. It is derived from the /wsa:ReferenceParameters /osci:TypeOfBusinessScenario associated to the WS-Addressing SOAP header element /wsa:To of the message.

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

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

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

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

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

Sequence of according header block elements of selected messages. For number of sequence entries see /oci:MsgAttributes above.

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

The normative outline for a MsgBoxGetNextRequest:

```xml
<s12:Envelope ...>
  <s12:Header ...>
    <wsa:Action>
      http://www.osci.eu/ws/2008/05/transport/urn/messageTypes/MsgBoxGetNextRequest
    </wsa:Action>
    <wsa:MessageID>xs:anyURI</wsa:MessageID>
    <wsa:To>xs:anyURI</wsa:To>
    <oci:TypeOfBusinessScenario wsa:IsReferenceParameter="1">xs:anyURI</oci:TypeOfBusinessScenario>
    ...
  </s12:Header>
  <s12:Body ...>
    <oci:MsgBoxGetNextRequest MsgBoxRequestId="xs:anyURI">
      <oci:LastMsgReceived> wsa:MessageID /oci:LastMsgReceived *
    </oci:MsgBoxGetNextRequest>
  </s12:Body>
</s12:Envelope>
```
Description of normative constraints on the outline listed above:

1647  /s12:Envelope/s12:Header/wsa:Action
1648  The value indicated herein MUST be used for that URI.
1649  /s12:Envelope/s12:Header/wsa:MessageID
1650  The request MUST carry a unique WS-Addressing MessageID.
1651  /s12:Envelope/s12:Header/wsa:To
1652  The address of the MsgBox (request destination) endpoint.
1653  /s12:Envelope/s12:Header/oci:TypeOfBusinessScenario
1654  The corresponding value of the initial MsgBoxFetchRequest or MsgBoxStatusListRequest
1655  MUST be supplied for this message type.
1656  /s12:Envelope/s12:Header/oci:TypeOfBusinessScenario/
1657  @wsa:IsReferenceParameter
1658  According to WS-Addressing, the element MUST be attributed with
1659  @wsa:IsReferenceParameter="1"
1660  The body of this message contains the actual
1662  /s12:Envelope/s12:Body/oci:MsgBoxGetNextRequest/@MsgBoxRequestID
1663  This attribute of type xs:anyURI MUST be provided with the value of the foregoing
1664  MsgBoxResponse/@MsgBoxRequestID. The MsgBox service MUST use it to correlate
1665  this MsgBoxGetNextRequest to the initial MsgBoxFetchRequest respective
1666  MsgBoxStatusListRequest.
1667  /s12:Envelope/s12:Body/oci:MsgBoxGetNextRequest/oci:LastMsgReceived *
1668  These optional elements of type wsa:AttributedURIType MAY be provided, when the
1669  underlying initial request was of type MsgBoxFetchRequest. The requestor SHOULD
1670  provide here the value(s) of the /wsa:MessageID of the last message(s) he received in
1671  the body of the foregoing response(s) to commit successful reception of those messages.
1672  This has to be realized as "reception acknowledged by requester" by the MsgBox
1673  instance: If the SOAP header element .../oci:MsgTimeStamps/oci:Reception is
1674  absent or the value of the SOAP header element
1675  .../oci:MsgTimeStamps/oci:Reception is zero, the actual server time of the
1676  MsgBox instance MUST now be set here and the value has to be signed according to
1677  chapter [8.1]. The resulting changes in the SOAP header block /oci:MsgTimeStamps
1678  MUST now be persisted in the MsgBox store.
1679  Upon receipt and authentication of this message, the MsgBox service MUST – depending on type of
1680  initial request referenced by /oci:MsgBoxGetNextRequest/@MsgBoxRequestID
1681  – deliver a MsgBoxResponse with the next message of the list indicated by
1682  /oci:MsgBoxResponse/@MsgBoxRequestID in the body of the response (rules denoted
1683  in chapter [8.2.3.1] apply)
1684  – deliver a MsgBoxResponse with the next portion of a /oci:MsgStatusList indicated by
1685  /oci:MsgBoxResponse/@MsgBoxRequestID in the body of the response (rules denoted
1686  in chapter [8.2.3.1] apply).
1687  Inside the SOAP header the element .../oci:MsgBoxResponse, choice .../oci:ItemsPending
1688  MUST be set to the actual value. If .../oci:ItemsPending becomes the value zero, this fact
1689  signals the requestor that the MsgBox instance may have discarded the search result list referenced
1690  by the identifier /oci:MsgBoxResponse/@MsgBoxRequestID.
8.2.5 MsgBoxCloseRequest

The functionalities of this message type are:

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

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

The normative outline for the MsgBoxCloseRequest:

```xml
<s12:Envelope ...>
  <s12:Header ...
    <wsa:Action>
      http://www.osci.eu/ws/2008/05/transport/urn/messageTypes(MsgBoxCloseRequest
    </wsa:Action>
    <wsa:MessageID>xs:anyURI</wsa:MessageID>
    <wsa:To>xs:anyURI</wsa:To>
    <osci:TypeOfBusinessScenario wsa:IsReferenceParameter="1">
      xs:anyURI
    </osci:TypeOfBusinessScenario>
  </s12:Header>
  <osci:MsgBoxCloseRequest MsgBoxRequestID="xs:anyURI">
    <osci:LastMsgReceived>wsa:MessageID</LastMsgReceived> *
  </osci:MsgBoxCloseRequest>
</s12:Envelope>
```

Description of normative constraints on the outline listed above:

/s12:Envelope/s12:Header/wsa:Action

The value indicated herein MUST be used for that URI.

/s12:Envelope/s12:Header/wsa:MessageID

The request MUST carry a unique WS-Addressing MessageID.

/s12:Envelope/s12:Header/wsa:To

The address of the MsgBox (request destination) endpoint.

/s12:Envelope/s12:Header/osci:TypeOfBusinessScenario

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


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


The body of this message contains the actual

/s12:Envelope/s12:Body/osci:MsgBoxCloseRequest/@MsgBoxRequestID

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

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

These optional elements of type wsa:AttributedURIType MAY be provided, when the
underlying initial request was of type MsgBoxFetchRequest. The requestor SHOULD
provide here the value(s) of the /wsa:MessageID of the last message(s) he received in
the body of the foregoing response(s) to successfully commit reception of those
messages. This has to be realized as "reception acknowledged by requester" by the
MsgBox instance: If the SOAP header element

.../osci:MsgTimeStamps/osci:InitialFetch is absent or present with the value
zero, the actual server time of the MsgBox instance MUST now be set here and the value
has to be signed according to chapter [8.1]. The resulting changes in the SOAP header
block /osci:MsgTimeStamps MUST now be persisted in the MsgBox store.

It should be noted, that this message type MUST be sent to the MsgBox instance always
when a MsgBoxFetchRequest was the initial message sent and the requestor pulled a
message successfully the first time (a new message). This triggers the commitment of the
SOAP header /osci:MsgTimeStamps/osci:Reception and

/osci:MsgTimeStamps/osci:InitialFetch time instances. It is up to
implementations of recipient instances, how to distinguish between "new" and already
processed messages, because the recipient transport gateway has no implicit control on
the state of the successful body processing (for example to mark message as "read" or
"processed" – this at least is under the control of the application targeted by the
message).

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

8.2.6 Processing Rules for MsgBoxGetNext/CloseRequest

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

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

Fault 9: MsgBoxRequestWrongReference

[Code] Sender
[Subcode] MsgBoxRequestWrongReference
[Reason] MsgBoxRequestID unknown or timed out.

8.3 Receipts

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

Besides provableness of what has been delivered / received when, for messages exchange patterns
using the MsgBox service it may be of interest for the Initiator to be informed, when the intended
recipient pulls the message from his MsgBox. More concrete – the business-scenario-needs of an
asynchronous message are bound to reaction times. In this case, a service requestor has to have
control to in-time delivery to the targeted recipient. In doubt of recipient activity concerning the request,
a service requestor (or even responder) may choose other communication channels to get in contact.
As there may be non-conformant implementations which don’t answer to a requested ReceptionReceipt, for additional comfort of control whether a message has been pulled yet by the intended recipient, the construct of a **FetchedNotification** is foreseen, which alike described for receipts, can be demanded by initiator and recipient instances. If requested, the recipients MsgBox instance MUST deliver such a notification to the endpoint, the initiator specified in his message; contents are the SOAP header elements indicating source and destination of the message and the time instant when it is pulled by the intended recipient. No separate signature is foreseen for this notification. The FetchedNotification is delivered in the SOAP body of a separate osci:Request to the endpoint to be exposed in the demand for this FetchedNotification – which again in general should be the MsgBox of the requesting initiator or recipient node.

### 8.3.1 Demanding Receipts

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

#### 8.3.1.1 Demand for Delivery Receipt

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

```xml
<osci:DeliveryReceiptDemand wsu:Id="xs:ID"
  @s12:role="http://www.w3.org/2003/05/soap-envelope/role/next"
  @s12:mustUnderstand="true"|
  @qualTSPforReceipt="true"|
  @echoRequest="true"|
  @wsa:ReplyTo= wsa:EndpointReference />
</osci:DeliveryReceiptDemand>
```

Description of elements and attributes in the schema overview above:

- **/osci:DeliveryReceiptDemand**
  - Optional SOAP header for indicating requirements for a DeliveryReceipt. It MUST be provided under the conditions mentioned above.

- **/osci:DeliveryReceiptDemand/@wsu:Id**
  - This attribute of type wsu:Id SHOULD be provided so that unambiguous references can be applied to this element.

- **/osci:DeliveryReceiptDemand/@s12:role**
  - This attribute of type xs:anyURI MAY be provided. It defaults to the URI outlined above. If this attribute is provided, it MUST be set to this value. According to the semantics of [SOAP12], this SOAP header block is designated to the next SOAP-node passed on the message route.

- **/osci:DeliveryReceiptDemand/@s12:mustUnderstand**
  - This Boolean attribute SHOULD be provided with the value "true". Following the semantics of [SOAP12], this SOAP header block MUST be understood and processed by the next SOAP-node passed on the message route willing to act in the role denoted by the foregoing attribute /osci:ReceiptDemand/@s12:role. For interoperability reasons with web service implementations not able to process the receipts defined here, it may be set to "false" or not present (which is equivalent to the value "false").
This optional Boolean attribute signals – if set to "true" – a qualified timestamp for the requested receipt information. If such a service is not available on the node, the receipt is demanded from, a fault (see chapter [8.3.2]) MUST be generated to the requesting node and the incoming message MUST be discarded.

This optional Boolean attribute signals – if set to "true" – that the requesting node requires the retransmission of the whole message in the required receipt. In this case, the node the receipt is demanded from, MUST provide the whole message in binary format in the receipt part of the response message (see chapter [8.3.2.1]). Care should be taken to use this feature with regard to caused overhead and bandwidth consumption.

If absent, this attribute defaults to "false".

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

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

<osci:TypeOfBusinessScenario>

www.osci.eu/ws/2008/05/common/urn/messageTypes/Receipt
</osci:TypeOfBusinessScenario>.

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

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

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

8.3.1.2 Demand for ReceptionReceipt

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

/osci:ReceptionReceiptDemand

Optional SOAP header for indicating requirements for a ReceptionReceipt.

/osci:ReceptionReceiptDemand/@wsu:Id

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

/osci:ReceptionReceiptDemand/@s12:role

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

/osci:ReceptionReceiptDemand/@s12:mustUnderstand

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

/osci:ReceptionReceiptDemand/@qualTSPforReceipt

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

/osci:ReceptionReceiptDemand/@echoRequest

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

/osci:ReceptionReceiptTo/wsa:ReplyTo

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

```xml
<osci:ReceptionReceiptDemand wsu:Id="...">
  @s12:role=
  "http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver" ?
  @s12:mustUnderstand= "true" | "false" ?
  @qualTSPforReceipt="true" | "false" ?
  @echoRequest= "true" | "false") ?
</osci:ReceptionReceiptDemand>
```
In case of delivering a receipt to a MsgBox instance, this is the default value for separating receipt message types from other ones (see chapter [6, Addressing Endpoints]).

### 8.3.2 Receipt Format and Processing

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

#### 8.3.2.1 Delivery Receipt

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

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

```xml
<osci:DeliveryReceipt @wsu:Id="xs:ID"
osci:ReceiptInfo
  @wsu:Id="...
osci:ReceiptIssuerRole=
    "http://www.osci.eu/ws/2008/05/transport/role/MsgBox" |
    "http://www.osci.eu/ws/2008/05/transport/role/Recipient" |
    "http://www.osci.eu/ws/2008/05/transport/role/Sender"
osci:ReceiptIssuerRole="http://www.osci.eu/ws/2008/05/transport/role/Relay"
  <wsa:MessageID>xs:anyURI</wsa:MessageID>
osci:MsgTimeStamps/>
<wsa:RelatesTo/> *
osci:To> wsa:EndpointReference </osci:To>
osci:ReplyTo> wsa:EndpointReference </wsa:ReplyTo>
osci:From> wsa:EndpointReference ?
osci:RequestEcho> xs:base64Binary </RequestEcho> ?
<osci:ReceiptInfo>
<ds:Signature/>
<osci:DeliveryReceipt>

Description of elements and attributes in the schema overview above:

/osci:DeliveryReceipt

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

/oci:DeliveryReceipt/@wsu:Id

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

/oci:DeliveryReceipt/osci:ReceiptInfo

Container to hold the receipt details.

/oci:DeliveryReceipt/osci:ReceiptInfo/ws@Id

This attribute of type xs:ID MUST be provided; the element must be referenceable from the signature element described below.
This element of type xs:anyURI MUST be provided with one of the URIs outlined above. The concrete value MUST expose the role of the receipt issuing node. If an osci:Request is targeted to a MsgBox instance, the value MUST be "http://www.osci.eu/ws/2008/05/transport/role/MsgBox". If an osci:Request is targeted directly to the recipients OSCI Gateway or an osci:Response message containing a demand for a DeliveryReceipt, the value MUST be "http://www.osci.eu/ws/2008/05/transport/role/Recipient".

If the receipt issuing node is a sender initially submitting the message, the value MUST be "http://www.osci.eu/ws/2008/05/transport/role/Sender"; if an active intermediary is just enriching and relaying the message, the value MUST be "http://www.osci.eu/ws/2008/05/transport/role/Relay".

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

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

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

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

If present in the message to be receipted, the /wsa:From SOAP header block of the message to be received.

The /wsa:From SOAP header block of the message to be received.

This element MUST be included, if the demand for the receipt contains the attribute /osci:DeliveryReceiptDemand/@echoRequest set to "true". The complete incoming message MUST be placed in this element in base64Binary format.

To be able to proof what has been sent, the initiator in this case is strongly advised to encrypt the message body for himself, too.

A digital signature of the DeliveryReceipt according to chapter [7.2].

One ds:Signature child element ds:Reference MUST points to the element /osci:DeliveryReceipt/ReceiptInfo using the same-URI reference mechanism via the ID-attribute of this element.

A second /ds:Signature/ds:Reference element MUST point to the /s12:Envelope/s12:Body block of the message to be receipted using the same-URI reference mechanism via the ID-attribute of the SOAP body block.
For a DeliveryReceipt, the received SOAP body block MUST to be signed "as is". The
actual server time in UTC-format MUST be provided in

\[\text{/osci:DeliveryReceipt/ds:Signature/ds:Object/}
\text{xades:QualifiyingProperties/xades:SignedProperties/}
\text{xades:SignedSignatureProperties/xades/SigningTime.}\]

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

\[\text{../ds:Signature/ds:Object/xades:QualifiyingProperties/}
\text{xades:UnsignedProperties /xades:UnsignedSignatureProperties/}
\text{xades:SignatureTimeStamp/xades:EncapsulatedTimeStamp}\]

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

Fault 10: QualTSPServiceNotAvailable

[Code] Sender
[Subcode] QualTSPServiceNotAvailable
[Reason] Requested qualified TSP service not provided by targeted node

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

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

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

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

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

### 8.3.2.2 Reception Receipt

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

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

```xml
<osci:ReceptionReceipt @wsu:Id="xs:ID" ? >
<osci:ReceiptInfo @wsu:Id="..." >
  <wsa:MessageID>x:s:ID</wsa:MessageID>
  <osci:MsgTimeStamps/>
  <osci:To> wsa:EndpointReference </osci:To>
  <wsa:ReplyTo> wsa:EndpointReference </wsa:ReplyTo>
  <wsa:From> wsa:EndpointReference </wsa:From> ?
  <osci:RequestEcho> xs:base64Binary </RequestEcho> ?
</osci:ReceiptInfo>
<ds:Signature/>
</osci:ReceptionReceipt>
```

Description of elements and attributes in the schema overview above:

- **/osci:ReceptionReceipt**
  - Container holding the child elements receipt data \(.../osci:ReceiptInfo\) and a `ds:Signature` element over \(.../osci:ReceiptInfo\).

- **/osci:ReceptionReceipt/@wsu:Id**
  - This attribute of type `xs:ID` SHOULD be provided so that unambiguous can be made to this `/osci:ReceptionReceipt` block.

- **/osci:ReceptionReceipt/osci:ReceiptInfo**
  - Container to hold the receipt details.

- **/osci:ReceptionReceipt/osci:ReceiptInfo/wsu:@Id**
  - This attribute of type `xs:ID` MUST be provided; the element must be referenceable from the signature element described below.

- **/osci:ReceptionReceipt/osci:ReceiptInfo/wsa:MessageID**
  - The `wsa:MessageID` SOAP header block of the message to be receipted.

- **/osci:ReceptionReceipt/isci:ReceiptInfo/isci:MsgTimeStamps**
  - The `/sci:MsgTimeStamps` SOAP header block; this element MUST be inserted after the receipting node that has inserted its specific timestamps according to chapter [8.1].

- **/osci:ReceptionReceipt/osci:ReceiptInfo/wsa:RelatesTo**
  - The `wsa:RelatesTo` SOAP header blocks of the message to be receipted.

- **/osci:ReceptionReceipt/osci:ReceiptInfo/wsa:From**
  - This element of type `wsa:EndpointReference` denotes the destination EPR of the message to be receipted. At least, it MUST contain the `wsa:To` SOAP header block of this message and those SOAP header blocks attributed by `@wsa:IsReferenceParameter="1"`.

- **/osci:ReceptionReceipt/isci:ReceiptInfo/wsa:ReplyTo**
  - If present in the message to be receipted, this is the `wsa:From` SOAP header block of the message to be receipted.

- **/osci:ReceptionReceipt/osci:ReceiptInfo/wsa:ReplyTo**
  - The `wsa:ReplyTo` SOAP header block of the message to be receipted.
This element MUST be included, if the demand for the receipt contains the attribute

`/osci:ReceptionReceiptDemand/@echoRequest` set to the value "true". The complete incoming message MUST be placed in this element in base64Binary format.

A digital signature of the ReceptionReceipt according to chapter [7.2.2]. The signature MUST be generated by the Ultimate Recipient after successful decryption of the whole SOAP body block. In case of a synchronous `osci:Response` to an `osci:Request` containing a ReceptionReceipt demand, this is the respective UltimateRecipient node on the initiator side. If complete decryption of the received SOAP body is not possible, a fault MUST be generated and further message processing MUST be aborted.

```
Fault 11: MsgBodyDecryptionError
[Code] Sender
[Subcode] MsgBodyDecryptionError
```

The fault [Details] property MAY outline – if known to the decrypting instance - the `ds:X509IssuerSerial` element of the certificate initially used for encryption.

One `ds:Signature` child element `ds:Reference` MUST point to the element `/osci:ReceptionReceipt/ReceiptInfo` using the same-URI reference mechanism via the ID-attribute of this element.

A second `/ds:Signature/ds:Reference` element MUST point to the `/s12:Envelope/s12:Body` element of the message to be receipted, using the same-URI reference mechanism via the ID-attribute of the SOAP body block. As already mentioned, the SOAP body block in advance MUST have been successfully decrypted.


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

```
.../ds:Signature/ds:Object/xades:QualifiyingProperties/
  xades:UnsignedProperties/
    xades:UnsignedSignatureProperties/
    xades:SignatureTimeStamp/xades:EncapsulatedTimeStamp
```

If no appropriate qualified TSP-service can be provided, a fault MUST be generated to the requestor instead of the ReceptionReceipt; processing of the incoming message MAY proceed (subject to policy to be defined for the concrete endpoint). See fault `QualTSPServiceNotAvailable` as defined in chapter [8.3.2.1]; the fault [Details] property MUST outline that this timestamp was requested for a ReceptionReceipt and it must state whether further message processing takes place or not.

The block `/osci:ReceptionReceipt` MUST be positioned as SOAP body of a new `osci:Request` message. This `osci:Request` message MUST be targeted to the endpoint denoted in

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

8.3.3 Submission and Relay Receipt

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

8.3.4 Fetched Notification

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

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

Description of elements and attributes in the schema overview above:

/osci:FetchedNotificationDemand

Header block containing the demand.

/osci:FetchedNotificationDemand/@wsu:Id

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

/osci:FetchedNotificationDemand/@s12:role

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

/osci:ReceiptTo/wsa:ReplyTo

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

```xml
<osci:TypeOfBusinessScenario>www.osci.eu/ws/2008/05/common/urn/mess
ageTypes/Notification/></osci:TypeOfBusinessScenario>
```

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

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

---

23 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."
Syntax for messages to deliver FetchedNotifications:

```xml
<s12:Envelope ...>
  <s12:Header ...>
    <wsa:Action>
      http://www.osci.eu/2008/05/transport/urn/messageTypes/OSCIRequest
    </wsa:Action>
    <wsa:MessageID>xs:anyURI</wsa:MessageID>
    <wsa:RelatesTo>xs:anyURI</wsa:RelatesTo>
    <wsa:To>xs:anyURI</wsa:To>
  </s12:Header>
  <s12:Body>
    <osci:FetchedNotification xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <osci:FetchedTime> xs:dateTime </osci:FetchedTime>
      <wsa:MessageID>xs:anyURI</wsa:MessageID>
      <wsa:To wsa:Address </wsa:To>
      <wsa:From wsa:EndpointReference </wsa:From>
    </osci:FetchedNotification>
  </s12:Body>
</s12:Envelope>
```

Description of elements and attributes in the schema overview above:

/s12:Envelope/s12:Header/wsa:Action
- The value indicated herein MUST be used for that URI.

/s12:Envelope/s12:Header/wsa:MessageID
- The message MUST carry a unique WS-Addressing MessageID.

/s12:Envelope/s12:Header/wsa:RelatesTo
- The message MUST carry the WS-Addressing MessageID for the message a FetchedNotification was requested for.

/s12:Envelope/s12:Header/wsa:To
- The address of the destination endpoint which was stated in the EPR of the request header element /osci:FetchedNotificationTo/wsa:ReplyTo/wsa:Address of the request message.

/s12:Envelope/s12:Header/osci:TypeOfBusinessScenario
- This is the instantiation of /wsa:ReferenceParameters bound to this EPR. It MUST be taken from the request header element /osci:FetchedNotificationTo/wsa:ReplyTo/wsa:ReferenceParameters of the request message.

@s12:Header/osci:TypeOfBusinessScenario/@wsa:IsReferenceParameter
- According to WS-Addressing, the element MUST be attributed with @wsa:IsReferenceParameter="1"

/s12:Envelope/s12:Body/osci:FetchedNotification
- Container holding the FetchedNotification.
8.3.5 Additional Receipt/Notification Demand Fault Processing Rules

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

For receipts, which have to be processed in asynchronous mode and/or delivered in a separate
osci:Request message, the receipt requestor has to be informed about possible fault occurrences
asynchronously. This MUST be done by placing the fault information in the body of an osci:Request
instead of the receipt block and delivered to the same endpoint the receipt message is expected at.
If the message to be receipted carries a wsa:FaultTo SOAP header block, this is the EPR the
osci:Request message carrying the fault MUST be targeted to. If this header is absent or – if present
and carrying the value http://www.w3.org/2005/08/addressing/anonymous in
wsa:FaultTo/Address, it MUST be targeted to the endpoint denoted in
/osci:ReceptionReceiptDemand/wsa:ReplyTo/Address. The according
wsa:ReferenceParameters SOAP header block MUST be

<osci:TypeOfBusinessScenario wsa:IsReferenceParameter="true">
  www.osci.eu/ws/2008/05/common/urn/messageTypes/Fault
</osci:TypeOfBusinessScenario>

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

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

NOTE: Message processing MUST NOT be aborted in these situations.24

8.3.5.1 Receipt Signature Validation

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

Fault 12: SignatureOfReceiptInvalid

<table>
<thead>
<tr>
<th>[Code]</th>
<th>Sender</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Subcode]</td>
<td>SignatureOfReceiptInvalid</td>
</tr>
<tr>
<td>[Reason]</td>
<td>Receipt signature verification failed.</td>
</tr>
</tbody>
</table>

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

24 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.
8.4 Message Meta Data

This specification version 2.0.1 introduces a new optional SOAP header block `osci21:MessageMetaData` containing information related to transport and meta data describing the message payload carried in the SOAP body.

**Note:** This header block must only be handled by OSCI version 2.0 aware implementations according to the rules described in this section.

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

**Note:** In future versions of OSCI Transport, these headers will be withdrawn. For downwards compatibility, they MUST be provided and processed according to details described below. If end entities provide an element `osci21:MessageMetaData`, version 2.01 aware implementations MUST copy related entries to headers mentioned above and maintain timestamps semantically identical in both headers; processing of receipts still MUST be done based on the information contained in the receipts/notification demand headers.

8.4.1 Re-used Type Definitions

Prior to the description of `osci21:MessageMetaData` below, the following sections specify types used for

- modelling logical identifiers/addresses and few, often used additional attributes for end entities

and

- generic modelling of diverse properties assertable to a message,

8.4.1.1 Non-empty String and URI

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

```xml
<xs:simpleType name="NonEmptyStringType">
  <xs:restriction base="xs:string">
    <xs:minLength value="1"/>
  </xs:restriction>
</xs:simpleType>
```

```xml
<xs:simpleType name="NonEmptyURIType">
  <xs:restriction base="xs:anyURI">
    <xs:minLength value="1"/>
  </xs:restriction>
</xs:simpleType>
```

8.4.1.2 Type Definitions used for Logical Addressing

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

---

25 Proposal will be submitted for a planned open consultation process
Commonly, application scenarios use different participant identifier schemes, which then are base for specific mapping / address resolution and routing details; a specific scheme may even determinate a channel/protocol to be chosen for data exchange.

The `osci21:PartyIdentifierType` below basically models a unique party by an identifier value, attributed by a type outlining the scheme of this value. Optional additional attributes carry informational items, not used for addressing purposes.

```xml
<xs:complexType name="PartyIdentifierType">
  <xs:annotation>
    <xs:documentation>Value of generic party identifier, as classified by @type attribute, e.g.: Prefix:Kennung</xs:documentation>
  </xs:annotation>
  <xs:simpleContent>
    <xs:extension base="xs:normalizedString">
      <xs:attribute name="type" type="xs:QName" use="required">
        <xs:annotation>
          <xs:documentation>Orientat: ebMS Core: type, how to interpret Party-Id value, e.g.: xöv oder Justiz</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="name" type="osci21:NonEmptyStringType">
        <xs:annotation>
          <xs:documentation>Optional "friendly name" value for displaying in user agents (as e.g. known from eMail)</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="category" type="xs:QName">
        <xs:annotation>
          <xs:documentation>Concrete role of party in business scenario (e.g. "buyer", "Meldebehörde", "Standesamt"...)</xs:documentation>
        </xs:annotation>
      </xs:attribute>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
```

Description of the type definition above:

1. `osci21:PartyIdentifierType` - `xs:normalizedString` carrying the party identifier value, extended by the attributes below.
2. `osci21:PartyIdentifierType/@type` - `xs:QName` outlining the scheme of this identifier value. QNames to be specified by scenarios implementing this specification based on this specific @type value.
3. `osci21:PartyIdentifierType/@name` - Optional non-empty `xs:string` intended to carry a "friendly name" of the party instance for e.g. displaying purposes in user agents.
4. `osci21:PartyIdentifierType/@category` - Optional non-empty `xs:QName` intended to carry a role of the party instance in a specific business-/communication-scenario. Definition of values of @category is out of scope of this specification.

NOTE for implementations aware of logical addressing:

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

Fault 13: UnknownPartyIdentifierType

[Code] Sender
[Subcode] UnknownPartyIdentifierType
[Reason] Party identifier type not known for actual subelement name in
osci21:MessageMetaData>

If @type attribute value is known and faults occur during resolution of the logical identifier
(e.g. directory lookup technical error or value of oci21:PartyIdentifierType not
found during lookup), a fault MUST be generated to the requestor (author or reader) and
processing of the incoming message MUST be aborted.

Fault 14: PartyIdentifierResolutionFault

[Code] Sender (if identifier not found); Receiver (if technical error)
[Subcode] PartyIdentifierResolutionFault
[Reason] Party identifier not found or technical error when resolving <actual subelement
name in oci21:MessageMetaData>
[Details] Related response of service used SHOULD be given here.

A further complex oci21:PartyType is provided, extending the type above by an optional binary
security token. This may be needed by scenarios dealing with end entity authentication on application
level.

Description of the type definition above:

/oci21:PartyType
   xs:sequence carrying the elements below.
/oci21:PartyType/Identifier
   Element of type oci21:PartyIdentifierType as defined above.
/oci21:PartyType/wsse:BinarySecurityToken
   Optional element of type wsse:BinarySecurityTokenType as specified in [WSS]. If
   present, MUST carry a security token (e.g. SAML, X509) of the party instance.
/oci21:PartyType/wsse:BinarySecurityToken/@ValueType
   This optional attribute according to [WSS] MUST be provided, outlined the type of actual
token carried here.
This optional attribute according to [WSS] SHOULD default to #Base64Binary. For sake of interoperability, this token encoding SHOULD NOT be changed.

8.4.1.3 Property Type

Description of properties auf a message is based on a key/value principle, whereas the key name itself resides in a certain scheme. Specification of schemes, keys, values, and assigned semantics of properties is out of scope of this specification; code tables defined elsewhere may serve as one possible reference.

```xml
<xs:complexType name="PropertyType">
  <xs:simpleContent>
    <xs:extension base="oci21:NonEmptyStringType">
      <xs:attribute name="scheme" type="xs:anyURI" use="required">
        <xs:annotation>
          <xs:documentation>URN of code table or namespace of property denoted</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="name" type="xs:QName" use="required">
        <xs:annotation>
          <xs:documentation>Name of property (in scheme denoted)</xs:documentation>
        </xs:annotation>
      </xs:attribute>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
```

Description of the type definition above:

- `oci21:PropertyValue` carrying the property value, extended by the attributes below. Possibly needed casting of values to simple types different from xs:string should be described by property definitions and derived from the property @name value.

- Mandatory attribute of type `xs:anyURI`, outlining the URN of a code table, defining property values or the namespace of the according property definition.

- Mandatory attribute of type `xs:QName` outlining the name of this property in the @scheme denoted above.
8.4.2 Description of Message Meta Data Header

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

This header covers bricks visualized above, grouping different aspects of message meta information according to the annotations in Figure 9. All complex subelements carry information related to core transport and general payload depiction, except sub-element osci21:MessageProperties, which is intended to cover meta information, agreed upon in specific business scenarios, which in most cases would only be meaningful and processable inside these distinct scenarios.

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

To facilitate message streaming at nodes, the transport of a message is targeted to, the optional element osci21:MsgSize of type xs:positiveInteger SHOULD be set by the initially sending OSCI Gateway with the total message size in bytes.

8.4.2.1 Delivery Attributes

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

```xml
<osci21:DeliveryAttributes>
  <osci21:Origin> xs:dateTime </osci21:Origin>
  <osci21:InitialSend> xs:dateTime </osci21:InitialSend>
  <osci21:NotBefore> xs:dateTime </osci21:NotBefore>
  <osci21:ObsoleteAfter> xs:date </osci21:ObsoleteAfter>
  <osci21:Delivery> xs:dateTime </osci21:Delivery>
  <osci21:InitialFetch> xs:dateTime </osci21:InitialFetch>
  <osci21:Reception> xs:dateTime </osci21:Reception>
  <osci21:ServiceQuality> osci21:PropertyType </osci21:ServiceQuality>
  <osci21:ReceiptRequests/>
</osci21:DeliveryAttributes>
```
Description of elements contained in `osci21:DeliveryAttributes`:

**Note:** Elements of `osci21:DeliveryAttributes` MUST NOT be provided or changed by other nodes on the message path than described here.

- `/osci21:Origin`
  
  This element of type `xs:dateTime` MAY be provided by an author to denote the time a message is created by the author and targeted to the sender.

- `/osci21:InitialSend`
  
  This element of type `xs:dateTime` MUST be provided by a sender to denote the time a message is targeted to the receiver.

- `/osci21:NotBefore`
  
  This element of type `xs:dateTime` MAY be provided by an author to denote the time a sender shall initiate message delivery. Senders MUST NOT initiate delivery before; if the value is in the past, delivery MUST be initiated immediately.

- `/osci21:ObsoleteAfter`
  
  This element of type `xs:date` MAY be provided by an author to denote the date after which a message is to be seen as obsolete for delivery and/or consumption. If and how this information is handled by this endpoint this message is targeted to, is outlined in the policy of this endpoint; see chapter [10.2.2] for details.

  To ensure downward compatibility, the sender MUST copy the element to the corresponding one in `osci:MsgTimeStamps`.

- `/osci21:Delivery`
  
  This element of type `xs:dateTime` MUST be provided by a recipient or MsgBox node when accepting an incoming message and MUST be set to the value of the actual time.

  To ensure downward compatibility, the recipient or MsgBox node MUST copy the element to the corresponding one in `osci:MsgTimeStamps`.

- `/osci21:InitialFetch`
  
  This element of type `xs:dateTime` MUST be provided by a MsgBox node with the value of the actual MsgBox server time when an authorized recipient initially pulls the message from his MsgBox instance and commits the successful initial reception of this message.

  This SHOULD be done by a recipient after the first successful pulling of the message from his MsgBox.

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

  To ensure downward compatibility, a MsgBox node MUST copy the element to the corresponding one in `osci:MsgTimeStamps`.

- `/osci21:Reception`
  
  This element of type `xs:dateTime` MAY be set by a recipient to his actual server time when successfully accepting an incoming message, but it should be considered that the signature is invalidated which was applied over SOAP header and body elements by the...
message issuing instance. A recipient MAY copy the element to the corresponding one in
osci:MsgTimeStamps.

It MUST be set by a MsgBox node to his actual server time when the recipient commits
the reception of a message through a MsgBoxGetNextRequest or MsgBoxCloseRequest.

To ensure downward compatibility, a MsgBox node MUST copy the element to the
corresponding one in osci:MsgTimeStamps.

…/osci21:ServiceQuality ?

This element of type osci21:PropertyType is intended to carry information for a certain
transport profile with requirements regarding priority, confidentiality and other service level
aspects. Specification of according properties and their semantics actually is out of scope
of this specification.\(^{26}\)

Properties may be defined for specific business scenarios; according implementation
extensions must be made available in this case.

…/osci21:ReceiptRequests ?

Container carrying a sequence of optional empty elements to denote which types of
delivery receipts are required by the author respective sender of a message.\(^{27}\)

**NOTE:** In aberration to OSCI 2.0, OSCI 2.1 type of receipt requests provide no possibility
to request a qualified timestamp to be applied to the receipt as well as no demand to the
retransmit the whole initial message in the required receipt (as defined above in section
8.3.1). These variants proved to not being used in practice and thus will be dropped finally
in a future version of this specification.

\(^{26}\) Actually one of KoSIT work items; once details available, will be respected in a follow-up of this specification.

\(^{27}\) 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, theses settings MUST have precedence.
Empty element, to be set by author if sender shall provide a receipt upon successful submission of the message.

Empty element, to be set by author or sender, if active nodes on the message path forwarding the message shall provide a receipt upon successful relay.

Empty element, to be set by author if sender indicates demand for a DeliveryReceipt. For downward compatibility, sender MUST build up an according osci:DeliveryReceiptDemand header block, osci:DeliveryReceiptDemand/wsa:ReplyTo MUST be set to ...

Empty element, to be set by author if sender indicates demand for a FetchedNotification. To ensure downward compatibility, sender MUST build up an according osci:FetchedNotificationDemand header block, osci:FetchedNotificationDemand/wsa:ReplyTo MUST be set to ...

Empty element, to be set by author if sender indicates demand for a ReceptionReceipt. For downward compatibility, sender MUST build up an according osci:ReceptionReceiptDemand header block, osci:ReceptionReceiptDemand/wsa:ReplyTo MUST be set to ...

Element of type wsa:EndpointReference, to be set by the author. If the required DeliveryReceipt should be routed some specialized endpoint consuming receipts the EPR of this endpoint MUST be exposed here.

If absent, the value defaults to the one for the wsa:From header block.

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

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.
Description of elements contained in osci21:Originators:

Note: Elements of osci21:Originators here MUST NOT be provided or changed by other nodes on the message path than described here.

.../osci21:Author ?

This element of type osci21:PartyType MUST be provided by an author of a message respective reader when responding to it.

.../osci21:Sender ?

This element of type osci21:PartyType MAY be provided by a sender of a message respective receiver when responding to it and these nodes have different ones from author respective reader.

.../osci21:ReplyTo ?

This element of type osci21:PartyType MAY be provided by an author or a sender of a message respective reader or receiver when responding to it, when replying to an address different to the one outlined in author is intended.

8.4.2.3 Destinations

This complex element carries details about message destinations.

<osci21:Destinations>
  <osci21:Reader> osci21:PartyType </osci21:Reader>
  <osci21:OtherDestinations>
    <osci21:OtherReaders> osci21:PartyIdentifierType </osci21:OtherReaders> *
    <osci21:CcReaders> osci21:PartyIdentifierType </osci21:CcReaders> *
  </osci21:OtherDestinations> ?
</osci21:Destinations>

Description of elements contained in osci21:Destinations:

Note: Elements of osci21:Destinations here MUST NOT be provided or changed by other nodes on the message path than Initiators.

.../osci21:Reader

This element of type osci21:PartyType MUST be provided by an author of a message respective reader when responding to it. It identifies the destinations of this actual message (or responds to such).

.../osci21:OtherDestinations ?

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

.../osci21:OtherDestinations/OtherReaders *

These elements of type osci21:PartyIdentifierType MAY be provided by an initiator of a message respective response to it, to outline other addressed readers.

.../osci21:OtherDestinations/CcReaders *

These elements of type osci21:PartyIdentifierType MAY be provided by an initiator of a message respective response to it, to outline other addressed readers in "carbon copy" role.
8.4.2.4  **MsgIdentification**

This complex element carries the application level identification of the actual message and its relation to other ones. It refers to the following `osci21:ProcessIdentifierType`:

```
<xs:complexType name="ProcessIdentifierType">
    <xs:annotation>
        <xs:documentation>Process ID message is related to</xs:documentation>
    </xs:annotation>
    <xs:extension base="osci21:NonEmptyStringType">
        <xs:attribute name="ProcessName" type="osci21:NonEmptyStringType">
            <xs:annotation>
                <xs:documentation>Process may have a name, e.g. "order"</xs:documentation>
            </xs:annotation>
        </xs:attribute>
    </xs:extension>
</xs:complexType>
```

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

An optional attribute `@ProcessName` of type `osci21:NonEmptyString` may be used to outline a process name.

```
<osci21:MsgIdentification>
    <wsa:MessageID> wsa:AttributedURIType </wsa:MessageID>
    <osci21:In-Reply-To> wsa:AttributedURIType </osci21:In-Reply-to> *
    <osci21:ProcessRef> ?
        <osci21:Requester> osci21:ProcessIdentifierType </osci21:Requester> ?
        <osci21:Responder> osci21:ProcessIdentifierType </osci21:Responder> ?
    </osci21:ProcessRef> ?
</osci21:MsgIdentification>
```

Description of elements contained in `osci21:MsgIdentification`:

```
.../wsa:MessageID

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

.../osci21:In-Reply-To *

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

.../osci21:ProcessRef ?

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

.../osci21:ProcessRef/Requester ?

Optional element of type `osci21:ProcessIdentifierType`, outlining the process assignment on requestor side.
Optional element of type `osci21:ProcessIdentifierType`, outlining the process assignment on responder side.

8.4.2.5 Qualifier

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

```
<osci21:Qualifier>
  <osci21:Subject> xs:string </osci21:Subject> ?
  <osci21:BusinessScenario> xs:Qname </osci21:BusinessScenario>
  <osci21:Service> xs:anyURI </osci21:Service>
  <osci21:MessageType version="oci21:NonEmptyString"> xs:Qname
</osci21:MessageType>
</osci21:Qualifier>
```

Description of elements contained in `osci21:Qualifier`:

```
.../osci21:Subject
  Optional element of type `xs:string`, carrying informational text about this message (as known from e-mail "about")

.../osci21:BusinessScenario
  This element of type `xs:QName` MUST be provided. Its value denotes the related business scenario. Possible schemes, values and assigned semantics are out of scope of this specification; code tables defined elsewhere may serve as one possible reference.

.../osci21:Service
  This element of type `xs:anyURI` MUST be provided. It denotes a distinct service in a certain business scenario. Possible values are out of scope of this specification; they must be agreed upon per business scenario.

.../osci21:MessageType
  This element of type `xs:QName` MUST be provided. It denotes a the type of message or document carried in the payload; possible values should normally be bound to specific business scenarios. These values are out of scope of this specification; they must be agreed upon per business scenario.

.../osci21:MessageType/@Version ?
  This optional attribute of type `osci21:NonEmptyString` MAY be provided to outline a specific version of the value of `osci21:MessageType`

8.4.2.6 MessageProperties

This optional sequence of `osci21:Property` elements is intended to carry information qualifying the message payload in a manner defined inside certain business scenarios.

```
<osci21:Messageproperties>
  <osci21:Property scheme="xs:anyURI" name=" xs:Qname">
    <osci21:NonEmptyStringType>
  </osci21:Property> *
</osci21:Messageproperties>
```

© 2013 Koordinierungsstelle für IT-Standards / Coordination Office for IT Standards
Description of elements contained in osci21:MessageProperties:

.../osci21:Property

Optional elements of type osci21:NonEmptyString, carrying a property value.

.../osci21:Property/@scheme

An osci21:Property element MUST carry an attribute @scheme of type xs:anyURI, identifying the namespace of the property definition. The value MAY be an URN to identify code tables defined elsewhere for property key values.

.../osci21:Property/@name

An osci21:Property element MUST carry the property name in form of an attribute @name of type xs:QName.

8.5 X.509-Token Validation on the Message Route

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

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

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

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

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

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

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

8.5.1 X.509-Token Container

This chapter describes this optional custom header to carry those certificates. In addition to the token themselves, following information is carried, which has to be provided by source-/target applications per token-usage:
Where a certificate is used in the body (by IDREF); information may be useful at recipient side when parsing a message after SOAP body block decryption and grouping together derived body block parts with their respective certificates/validation results (at least. validation of signatures contained in the body SHOULD happen now)

- Application time instant (this is the time instant a certificate must be proven as valid)
- Possibility to indicate forced online OCSP request to downstream validation service nodes (force bypassing possible caching of once gained OCSP responses).

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

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

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

Syntax for an optional /osci:X509TokenContainer:

```xml
<osci:X509TokenContainer validateCompleted=("true" | "false")? >
  <osci:X509TokenInfo Id="xs:ID" validated=("true" | "false")? >
    <ds:X509Data>
      <ds:X509Certificate>
        ...<ds:X509Data>
      </ds:X509Certificate>
    </ds:X509Data>
  </osci:X509TokenInfo>
  <osci:TokenApplication ocspNoCache=("true" | "false")? >
    validateResultRef="xs:IDREF" ? >
      <osci:TimeInstant>xs:dateTime</osci:TimeInstant>
      <osci:MsgItemRef>xs:IDREF</osci:MsgItemRef>
    </oci:TokenApplication>
  </osci:X509TokenInfo>
</osci:X509TokenContainer>
```

Description of elements and attributes in the schema overview above:

/osci:X509TokenContainer

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

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

/oci:X509TokenContainer/@validateCompleted

This optional Boolean attribute MUST be provided with the value "true" by a validation processing node, when processing was successfully passed for all application instances of all contained items /osci:X509TokenInfo. It MUST NOT be provided with the value "true" if this condition is false – i.e. only partially successful validation processing. It MUST NOT be provided or changed by other logical instances than validation processing nodes.
If an \texttt{/osci:X509TokenContainer} is present, it MUST contain at least one item of this
type; content description follows here:

\texttt{/osci:X509TokenContainer/osci:X509TokenInfo/@Id}

This mandatory attribute of type \texttt{xs:ID} MUST be provided. As the whole
\texttt{/osci:X509TokenContainer} is initially to be generated by a source application
instance, the value must be a UUID; the UUID-value MUST NOT start with a character
unlike the \texttt{xs:ID} production rules and SHOULD therefore be preceded by a string of
"uuid:". This attribute MUST NOT be provided or changed by other logical instances than
source applications.

\texttt{/osci:X509TokenContainer/osci:X509TokenInfo/@validated} ?

This optional Boolean attribute of type \texttt{xs:ID} MUST be provided with a value of "true" by
a validation processing node, when processing was successfully passed for all application
instances of this item \texttt{/osci:X509TokenInfo}. It MUST NOT be provided with the value
"true" if this condition is false – i.e. only partially successful validation processing. It MUST
NOT be provided or changed by other logical instances than validation processing nodes.

\texttt{/osci:X509TokenContainer/osci:X509TokenInfo/ds:X509Data}

The X.509-Token of type \texttt{ds:Data} MUST be provided here by the source application
instance. Other sub-elements than \texttt{X509Certificate} foreseen in \texttt{ds:X509Data} MUST
NOT be provided.

\texttt{/osci:X509TokenContainer/osci:X509TokenInfo/ds:X509Data/ds:X509Certificate}

Subelement \ldots/\texttt{ds:509Certificate} MUST be provided. It MUST NOT be provided or
changed by other logical instances than source applications.

\texttt{/osci:X509TokenContainer/osci:X509TokenInfo/osci:TokenApplication}

A source application MUST initially provide this container containing application details of
the X.509-Token.

\texttt{/osci:X509TokenContainer/osci:X509TokenInfo/osci:TokenApplication/}
\@ocspNoCache ?

This optional Boolean attribute MUST be provided with a value of "true" by the source
application instance, when the downstream validation service node shall be forced
bypassing possible caching of OCSP responses while validating this certificate. If not
provided with the value "true", a validation service node MAY use caching mechanisms to
build up validation results. It MUST NOT be provided or changed by other logical
instances than a source application instance.

\texttt{/osci:X509TokenContainer/osci:X509TokenInfo/osci:TokenApplication/}
\@validateResultRef ?

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

\texttt{/osci:X509TokenContainer/osci:X509TokenInfo/osci:TokenApplication/}
\oscix007b:TimeInstant

This element of type \texttt{xs:dateTime} MUST be provided by the source application instance
and carry the token application time instant. This time instant MUST be taken as validation
time instant by the validation processing node (see next chapter). It MUST NOT be
provided or changed by other logical instances than source applications.
This element of type xs:IDREF MUST be provided by the source application instance and carry a reference to the cryptographic element in the message body where the token was used. It MUST NOT be provided or changed by other logical instances than source applications.

8.5.2 X.509-Token Validation Results

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

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

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

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

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

8.5.3 Verification of XKMS Validate Result Signatures

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

---

29 These extensions are subject to alignment in the context of running EU-wide "Large Scale Pilot" (lsp) projects. Concrete work on these issues is done by the project PEPPOL, see www.peppol.eu. One goal is a common 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.
Fault 15: **SignatureOfValidateResultInvalid**

[Code]  Sender

[Subcode] SignatureOfValidateResultInvalid

[Reason]  Verification failed for XKMS validate result

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

### 8.6 General Processing of Custom Header Faults

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

Fault 16: **MsgHeaderStructureSchemaViolation**

[Code]  Sender

[Subcode] MsgHeaderStructureSchemaViolation

[Reason]  One or more OSCI headers violate schema definitions

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

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

For a quick overview, constituents of each message type are illustrated in diagrams.30

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

30 All pictures have been revised in this specification version.
9.1 osci:Request

Figure 10: osci:Request header and body block assembly
SOAP header blocks:

/wst:RequestSecurityTokenResponseCollection ?

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

/wsse:Security

This header block MUST be present, carrying message protection data and initiator authentication and authorization information items according to the security policy of the node, the message is targeted to, see chapter [7.1] for details.

/osci:MsgTimeStamps ?

This optional header block MUST only be set by the initiator, if he wishes to supply a ...
/osci:ObsoleteAfter date in here. This header block MUST be set by a MsgBox instance and MAY be set – if not yet present - by a recipient instance. This header block MUST always be relayed, see chapter [8.1] for details.

/osci:DeliveryReceiptDemand ?

This optional header block MUST only be set by the initiator, if he wishes to receive a DeliveryReceipt in the backchannel response message. This header block MUST be removed from the message by the node processing it, see chapter [8.3.1.1] for details.

/osci:ReceptionReceiptDemand ?

This optional header block MUST only be set by the initiator, if he wishes to receive a ReceptionReceipt. This header block MUST be removed from the message by the node processing it, see chapter [8.3.1.2] for details.

/osci:FetchedNotificationDemand ?

This optional header block MUST only be set by the initiator, if he wishes to receive a FetchedNotification. This header block MUST only be processed by a MsgBox node instance and MUST be removed from the message after processing it, see chapter [8.3.4] for details.

/osci:X509TokenContainer ?

This optional header block SHOULD be provided by the initiator, if he wishes to enable certificate validation on the message route. This header block MUST always be relayed, see chapter [8.5.1] for details.

/xkms:CompoundResult *

These optional header blocks MUST be provided by nodes processing the header block /osci:X509TokenContainer. These header blocks containing /xkms:ValidateResult elements - MUST always be relayed, see chapter [8.5.2] for details.

/wsa:*

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

/osci21:MessageMetaData ?

For version 2.0.1 aware implementations, this optional header block MUST be provided by the source application, see chapter [8.4] for details. It MAY be provided informational for version 2.0 aware implementations. This header block MUST always be relayed.
9.2 **osci:Response**

Carries the request message ContentData, generally MUST be encrypted by the source application or initiator for the Ultimate Recipient.

![Diagram of osci:Response header and body block assembly](image)
SOAP header blocks:

/wst:RequestSecurityTokenResponseCollection

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

/wsse:Security

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

/osci:DeliveryReceipt

This optional header block MUST be provided by the responding endpoint, if a demand for a DeliveryReceipt is present in the corresponding request. This header block MUST not be discarded or changed on the message route, see chapter [8.3.2] for details.

/osci:ReceptionReceiptDemand

This optional header block MUST only be set by the responding recipient, if he wishes to receive a ReceptionReceipt for the response message. This header block MUST NOT be set by a MsgBox instance. This header block MUST be removed from the message by the node processing it, see chapter [8.3.1.2] for details.

/osci:X509TokenContainer

This optional header block SHOULD be provided by the responding recipient, if he wishes to enable certificate validation on the message route. This header block SHOULD NOT be set by a MsgBox instance. This header block MUST always be relayed, see chapter [8.5.1] for details.

/xkms:CompoundResult

These optional header blocks MUST be provided by nodes processing the header block /osci:X509TokenContainer. These header blocks - containing /xkms:ValidateResult elements - MUST always be relayed, see chapter [8.5.2] for details.

/wsa:*

All WS-Addressing headers MUST (if in continuous blocks) /MAY (if in dashed blocks) be supplied by the resending endpoint and MUST always be relayed, see chapter [6.1.2] for details.

/osci21:MessageMetaData

For version 2.0.1 aware implementations, this optional header block MUST be provided by the responding target application, see chapter [8.4] for details. It MAY be provided informational for version 2.0 aware implementations. This header block MUST always be relayed.

SOAP body:

May carry the response message ContentData in case of point-to-point scenarios. If present, it generally MUST be encrypted by the target application or recipient for the initiator. If an error occurred, a fault message is placed here instead.
9.3 MsgBoxFetchRequest

Figure 12: MsgBoxFetchRequest header and body block assembly

SOAP header blocks:

/wsse:Security

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

/wsa:*

All WS-Addressing headers MUST (if in continuous blocks) /MAY (if in dashed blocks) be supplied by the initiator, see chapter [6.1.2] and [8.2.1] for details.
SOAP body:
Carries the details of the MsgBoxFetchRequest, generally MUST be transport encrypted, see chapter [8.2.1] for details.

9.4 MsgBoxStatusListRequest

Figure 13: MsgBoxStatusListRequest header and body block assembly

SOAP header blocks:

/wsse:Security

This header block MUST be present, carrying message protection data and requestor (MsgBox owner in this case) authentication and authorization information items according the security policy MsgBox instance, see chapter [7.1] for details.
All WS-Addressing headers MUST (if in continuous blocks) /MAY (if in dashed blocks) be supplied by the initiator, see chapter [6.1.2] and [8.2.2] for details.

SOAP body:
Carries the details of the MsgBoxFetchRequest, generally MUST be transport encrypted, see chapter [8.2.2] for details.

9.5 MsgBoxResponse

Figure 14: MsgBoxResponse header and body block assembly

SOAP header blocks:

/\texttt{wsse:Security}

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

/\texttt{wsa:*}

All WS-Addressing headers MUST (if in continuous blocks) /MAY (if in dashed blocks) be supplied by the initiator, see chapter [6.1.2] and [8.2.3] for details.
This header, carrying status information concerning the actual message box access, MUST be set by the resending MsgBox instance, see chapter [8.2.3] for details.

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

9.6 MsgBoxGetNextRequest

Figure 15: MsgBoxGetNextRequest header and body block assembly

© 2013 Koordinierungsstelle für IT-Standards / Coordination Office for IT Standards
SOAP header blocks:

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

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

Carries the details of the MsgBoxGetNextRequest, generally MUST be transport encrypted, see chapter [8.2.4] for details.
## 9.7 MsgBoxCloseRequest

![Diagram of MsgBoxCloseRequest]

**Figure 16: MsgBoxClose header and body block assembly**

### SOAP header blocks:

**/wsse:Security**

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

**/wsa:***

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

Carries the details of the MsgBoxCloseRequest, generally MUST be transport encrypted, see chapter [8.2.5] for details.
10 Policies and Metadata of Communication Nodes and Endpoints

10.1 General Usage of Web Service Description Language

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

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

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

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

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

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

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

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

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

10.1.1 WSDL and Policies for MEP Synchronous Point-To-Point

For this communication scenario, a description of endpoint requirements and abilities SHOULD be outlined in one WSDL, containing all services and ports with their respective policies available here.

The following general requirements MUST be considered when designing WSDL instances:

- /wsdlll:port MUST always contain an entry /wsa:EndpointReference with the ...
  /wsa:Address element as well as ...
  /wsa:ReferenceParameters outlining the URI

---

31 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.
of the .../osci:TypeOfBusinessScenario served by this port32. Each specific
/osci:TypeOfBusinessScenario itself correlates to a concrete Content Data
message structure given by the /wsdl11:port reference chain to a /wsdl11:binding
and /wsdl11:portType entry in this WSDL instance.

10.1.2 WSDL and Policies for Asynchronous MEPs via Message Boxes

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

- Initially, a source application has to be built up the SOAP body content, according to a
  concrete schema bound to the actual underlying /osci:TypeOfBusinessScenario. In
  addition, security requirements bound to the recipient apply like end-to-end encryption and
digital signatures to be applied to Content Data, which SHOULD be expressed by according
WS Security Policy expressions.

- For transport to the recipient's MsgBox instance, the WSDL and policies of this target node
  apply. For every /osci:TypeOfBusinessScenario accepted here, the body structure is of
type xenc:EncryptedData. The WS Security Policy, if effect here, MUST NOT lead to
initiate body description processing, as the therefore needed private encryption key is only
known to the recipient node.

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

To solve this problem for this version of the OSCI Transport specification, the following
recommendation applies33:

- The MsgBox node exposes the WSDL and policies according to its needs on opaque body,
  transport security and authentication/authorization per accepted
/osci:TypeOfBusinessScenario.

- WSDL and policies in effect for the recipient node are referenced or contained in the
  /wsa:EndpointReference/wsa:Metadata element as described in chapter [6.1.1],
bound to the /wsdl11:port policy attachment point.

10.2 OSCI Specific Characteristics of Endpoints

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

10.2.1 Certificates used for Signatures and Encryption

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

- Encryption to be processed on initiator side
  
  o End-to-end encryption of Content Data targeted from a source application to a target
  application

  o Transport encryption, in cases where asymmetric encryption is required by a specific
  application scenario – in general expressed by an adequate security policy

32 Following chapter [6.1.1], use of WS-Addressing in OSCI
33 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.
Certificates used for signatures at recipient side (respective his MsgBox service); an initiator MAY – in cases of doubt - cross-check whether received signatures are generated with the certificates exposed in this endpoint policy (detection of possible man-in-the-middle attacks):

- Signature application for OSCI receipts and possible other message parts – in cases where the signature must be useable for long term provableness
- If offered: generation of cryptographic timestamps.

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

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

```
<wsp:Policy wsu:Id="xs:ID">
  <osci:X509CertificateAssertion>
    <wsp:ALL>
      <wsse:SecurityTokenReference wsu:Id="xs:ID" ?
        Usage="http://www.osci.eu/ws/2008/05/common/names/TokenUsage/e2eContentEncryption" |
        Usage="http://www.osci.eu/ws/2008/05/common/names/TokenUsage/TransportEncryption"
        |
        Usage="http://www.osci.eu/ws/2008/05/common/names/TokenUsage/ReceiptSigning" *
        osaci:Role="http://www.osci.eu/ws/2008/05/transport/role/Recipient" |
        osaci:Role="http://www.osci.eu/ws/2008/05/transport/role/MsgBox" |
        osaci:Role="http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver" * >
      </wsse:SecurityTokenReference>
    </wsp:ALL>
  </osci:X509CertificateAssertion>
</wsp:Policy>
```
Description of elements and attributes in the schema overview above:

3252 /wsp:Policy
3253 The whole assertion MUST be embedded in a policy block according to WS Policy.
3255 /wsp:Policy/@wsu:Id
3256 To be referenceable, the policy MUST carry an attribute of type xs:ID.
3257 /wsp:Policy/osci:X509CertificateAssertion
3258 The policy block containing all assertions.
3260 Following the semantics of WS Policy Framework [WSPF], all behaviours represented by
3261 the assertions embedded in this block are required/valid.
3263 This element defined in WS Security [WSS] MUST be used as container for a single
3264 X.509v3 certificate (or a reference to it) and its attributes.
3266 As all single certificate details are contained in this block, for brevity full path qualification
3268 is symbolized by [SingleToken] in the following descriptions.
3269 [SingleToken]/@wsu:id
3270 A certificate contained/described in this policy MUST be uniquely referenceable — i.e.,
3271 from other policies describing the same endpoint. This attribute of type xs:ID MUST
3272 carry an appropriate unique value.
3273 [SingleToken]/@Usage
3274 This attribute defines the purposes a certificate is used for, at least one of the URIs
3275 outlined above MUST be supplied as value in this attribute of type list of xs:anyURI.
3276 Predefined usage semantics are:

<table>
<thead>
<tr>
<th>Usage for</th>
<th>URI</th>
</tr>
</thead>
<tbody>
<tr>
<td>End-to-end encryption of Content Data</td>
<td><a href="http://www.osci.eu/ws/2008/05/common/names/TokenUsage/e2eContentEncryption">http://www.osci.eu/ws/2008/05/common/names/TokenUsage/e2eContentEncryption</a></td>
</tr>
<tr>
<td>Asymmetric transport encryption</td>
<td><a href="http://www.osci.eu/ws/2008/05/common/names/TokenUsage/TransportEncryption">http://www.osci.eu/ws/2008/05/common/names/TokenUsage/TransportEncryption</a></td>
</tr>
<tr>
<td>Signature of OSCI receipts; also applicable for signatures of other message parts</td>
<td><a href="http://www.osci.eu/ws/2008/05/common/names/TokenUsage/ReceiptSigning">http://www.osci.eu/ws/2008/05/common/names/TokenUsage/ReceiptSigning</a></td>
</tr>
<tr>
<td>Generation of cryptographic timestamps</td>
<td><a href="http://www.osci.eu/ws/2008/05/common/names/TokenUsage/TSPSigning">http://www.osci.eu/ws/2008/05/common/names/TokenUsage/TSPSigning</a></td>
</tr>
</tbody>
</table>

3277 Table 9: OSCI X.509-Token usages

3278 [SingleToken]/@osci:Role
3279 This attribute defines logical roles a certificate is assigned to, for one of the URIs outlined
3280 above a value MUST be supplied in this attribute of type list of xs:anyURI. Regularly, a
single certificate SHOULD NOT be assigned to more than one role; as constellations are imaginable, where logical roles are pooled – like for a recipient and Ultimate Recipient, which are using the same signature certificate - in these cases more than one role assignment MAY be used.

For example, this role attribute allows initiators to control, whether a receipt is signed with the right certificate used by the specific receipt issuer role outlined in the receipt.

Predefined logical roles are:

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

Table 10: SOAP/OSCI roles assigned to token usages

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

Choice for embedded tokens

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

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

Generic container to carry security tokens in binary format; MUST contain the X.509v3 certificate in base64Binary format.

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

As only X.509v3 certificates are described/contained here, the URI outlined above MUST be supplied as value in this attribute of type of xs:anyURI.
Hence X.509v3 certificates MUST be encoded in base64Binary format here, the URI outlined above MUST be supplied as value in this attribute of type of `xs:anyURI`.

**Choice for directly referencing tokens**

This choice MUST be taken, if referencing X.509v3 certificates stored otherwise.

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

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

**Choice for referencing tokens by thumbprint**

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

This choice MUST be taken if referencing X.509v3 certificates by thumbprint. Other choices foreseen by WS-Security for `/wsse:KeyIdentifier` MUST NOT be used.

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

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

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

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

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

**10.2.2 Endpoint Services and Limitations**

OSCI recipients respective MsgBox services MAY offer/expose the following services and limits:

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

Syntax for OSCI endpoint services policy assertions:

```xml
<wsp:Policy wsu:Id="xs:ID">
  <osci:QualTSPAssertion PolicyRef="xs:anyURI"/>
  <osci:ObsoleteAfterAssertion PolicyRef="xs:anyURI"/>
  <osci:MsgLimitsAssertion>
    <osci:MaxSize/>
    <osci:MaxPerHour/>
  </osci:MsgLimitsAssertion>
</wsp:Policy>
```

Description of elements and attributes in the schema overview above:

-/wp:Policy
  The whole assertion MUST be embedded in a policy block according to WS Policy.
-/wp:Policy/@wsu:Id
  To be referenceable, the policy MUST carry an attribute of type xs:ID.
-/wp:Policy/osci:QualTSPAssertion
  The presence of this element signals the availability of a qualified timestamp service.
  This optional attribute of type xs:anyURI SHOULD be provided and should carry a link to i.e. human readable policies, describing terms and conditions under which this service is made available.
-/wp:Policy/osci:ObsoleteAfterAssertion
  The presence of this element signals the fact that this endpoint will care about a SOAP header entry /osci:MsgTimeStamps/osci:ObsoleteAfter.
-/wp:Policy/osci:ObsoleteAfterAssertion/@PolicyRef
  This optional attribute of type xs:anyURI MAY be provided and carry a link to i.e. human readable policies describing terms and conditions about deletion of messages, marked to be obsolete meanwhile.
-/wp:Policy/osci:ObsoleteAfterAssertion/MsgRetainDays
  This optional element of type xs:positiveInteger SHOULD be provided to expose the number of days, a message is still hold available after the date provided by the .../osci:ObsoleteAfter entry.
This optional element of type `xs:nonNegativeInteger` SHOULD be provided to expose the number of days a warning is generated before the date provided by the `.../osci:ObsoleteAfter` entry. Thus, an escalation procedure could be triggered for messages seen to be of high importance. How this warning is generated and delivered is a matter of implementation of this service and SHOULD be described in the terms and conditions policy.34

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

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

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

```
Fault 17: MsgSizeLimitExceeded
[Code]  Sender
[Subcode] MsgSizeLimitExceeded
[Reason] Message size exceeds policy
```

This optional element of type `xs:positiveInteger` outlines the maximum amount of messages accepted per hour from the same originating node35.

If an incoming message that originates from the same targeting node, exceeds this limit, the message MUST be withdrawn and a fault MUST be returned to the targeting node:

```
Fault 18: MsgFrequencyLimitExceeded
[Code]  Sender
[Subcode] MsgFrequencyLimitExceeded
[Reason] Message frequency per hour exceeds policy
```

10.3 WS Addressing Metadata and WS MakeConnection

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

The following policy assertions MUST be bound to the `wsl11:port` (WSDL 2.0: endpoints) or `wsl11:binding` endpoint policy subjects, which accept messages of type `osci:Request`; WS MakeConnection is not supported in this case:

```
<wsp:Policy wsu:Id="xs:ID" ?>
  <wsam:Addressing>
    <wsp:Policy/> ?
  </wsam:Addressing>
</wsp:Policy>
```

34 This warning could e.g. be delivered in the body of an `osci:Request` to the initiator alike the FetchedNotification message.

35 No further details defined here, it is left to implementations how to define appropriate count starting and reset points.
This policy ascertains the use of WS-Addressing and that the endpoint requires request messages to use response endpoint EPRs that contain something other than the anonymous URI as the value in the SOAP header element `/wsa:ReplyTo/wsa:Address`.

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

```
<wsp:Policy wsu:Id="xs:ID"/>
<wsp:Policy>
  <wsam:Addressing>
    <wsam:AnonymousResponses/>
    <wsam:Addressing/>
  </wsp:Policy>
</wsp:Policy>
```

This policy ascertains the use of WS-Addressing and that the endpoint requires request messages to use response endpoint EPRs that carry an URI value of "http://www.w3.org/2005/08/addressing/anonymous" in the SOAP header element `/wsa:ReplyTo/wsa:Address`.

### 10.4 WS Reliable Messaging Policy Assertions

Support of WS Reliable Messaging is strongly recommended for OSCI version 2 conformant implementations. Adequate policy assertions MUST be used to ascertain the details of reliable messaging exchange. We refer to the specification WS Reliable Messaging Policy Assertions Version 1.1 [WSRMP] in this point, with no further profiling.

### 10.5 MTOM Policy Assertion

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

```
<wsp:Policy wsu:Id="xs:ID"/>
<wspmtom:OptimizedMimeSerialization/>
</wsp:Policy>
```

We refer to the specification draft [MTOMP].
10.6 WS Security Profile and Policy Assertions

10.6.1 Endpoint Policy Subject Assertions

The bindings, outlined in this chapter, apply for transport level encryption and signature\(^{36}\).

10.6.1.1 Symmetric Binding

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

According to [WSSP], this assertion SHOULD apply to the endpoint policy subject \texttt{wsdl11:binding}; it MAY apply to operation policy subject \texttt{wsdl11:binding/wsdl11:operation}.

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

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

\textbf{R1230} - This profiling restricts the usage of \texttt{wssp:ProctectionToken}; distinct \texttt{wssp:EncryptionToken} and \texttt{wssp:SignatureToken} MUST NOT be used.

10.6.1.2 Asymmetric Binding

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

According to [WSSP], this assertion SHOULD apply to the endpoint policy subject \texttt{wsdl11:binding}; it MAY apply to operation policy subject \texttt{wsdl11:binding/wsdl11:operation}.

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

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

Fault 19: \texttt{EncryptionCertNotValid}

<table>
<thead>
<tr>
<th>[Code]</th>
<th>Sender</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Subcode]</td>
<td>EncryptionCertNotValid</td>
</tr>
<tr>
<td>[Reason]</td>
<td>Encryption certificate not stated to be valid</td>
</tr>
</tbody>
</table>

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

In the context where certificates are used by a recipient or his MsgBox node (as described in the chapter [10.2.1]), the assertions \texttt{/wssp:RecipientEncryptionToken} and \texttt{/wssp:RecipientSignatureToken} SHOULD point to the according certificate entries in the recipients metadata file.

---

\(^{36}\) Note that for end-to-end encryption of content data a hybrid technique as defined in [7.3.1] must be used.
10.6.1.3 Transport Binding

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

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

This assertion MUST apply to the endpoint policy subject wsdl11:binding.

10.6.2 Message Policy Subject Assertions

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

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

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

Required message parts policy assertion:

```xml
<wsp:Policy>
  <wsp:ExactlyOnce>
    <wsp:ALL>
      <wssp:RequiredParts xmlns:wssp="..." ... >
        <wssp:Header Name="xs:NCName"? Namespace="xs:anyURI" ... / > +
      </wssp:RequiredParts>
    </wsp:ALL>
  </wsp:ExactlyOnce>
</wsp:Policy>
```

Signed message parts policy assertion:

```xml
<wsp:Policy>
  <wsp:ExactlyOnce>
    <wsp:ALL>
      <wssp:SignedParts xmlns:wssp="..." ... >
        <wssp:Body />
        <wssp:Header Name="xs:NCName"? Namespace="xs:anyURI" ... / > +
      </wssp:SignedParts>
    </wsp:ALL>
  </wsp:ExactlyOnce>
</wsp:Policy>
```

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

Encrypted message parts policy assertion:

```xml
<wsp:Policy>
  <wsp:ExactlyOnce>
    <wsp:ALL>
      <wssp:EncryptedParts xmlns:wssp="..." ... >
        <wssp:Body />
        <wssp:Header Name="xs:NCName"? Namespace="xs:anyURI" ... / > *
      </wssp:EncryptedParts>
    </wsp:ALL>
  </wsp:ExactlyOnce>
</wsp:Policy>
```

If potentially unsecured network connections are used for message exchange, the following requirement applies:
If the Content Data carried in the SOAP body is not encrypted end-to-end, the body block MUST be transport encrypted.

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

These policy assertions MUST be bound to the message policy subject:

wsdl11:binding/wsdl11:operation/wsdl11:input

respective

wsdl11:binding/wsdl11:operation/wsdl11:output

### 10.6.3 Algorithm Suite Assertions

In the chapters [7.2.1 and 7.3.2], restrictions are defined to suitable cryptographic algorithms, which leads to the following restrictions\(^\text{37}\):

For the symmetric case, the following restriction applies for the algorithm suite assertion:

```
<wsp:Policy>
  <wssp:AlgorithmSuite>
    <wsp:Policy>
      (<wssp:Basic256Sha256 ... /> | <wssp:Basic192Sha256 ... /> | <wssp:Basic128Sha256 ... /> | <wssp:TripleDesSha256 ... /> )
    </wsp:Policy>
  </wssp:AlgorithmSuite>
</wsp:Policy>
```

For the asymmetric case, the following restriction applies for the algorithm suite assertion:

```
<wsp:Policy>
  <wssp:AlgorithmSuite>
    <wsp:Policy>
      (<wssp:Basic256Sha256Rsa15 ... /> | <wssp:Basic192Sha256Rsa15 ... /> | <wssp:Basic128Sha256Rsa15 ... /> | <wssp:TripleDesSha256Rsa15 ... /> )
    </wsp:Policy>
  </wssp:AlgorithmSuite>
</wsp:Policy>
```

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

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

\(^{37}\) 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 reposible OASIS TC and completed as soon as possible by corrigenda for this document.
11 Applying End-to-end Encryption and Digital Signatures on Content Data

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

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

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

- SignRequest
- VerifyRequest
- EncryptRequest
- DecryptRequest.

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

To use the OSCI feature of certificate validation on the message route, messages producing instances SHOULD supply certificates used for cryptographical operations on Content Data level in a structure described as "X.509-Token Container" in chapter [8.5.1]. This container must be carried in a message as custom SOAP header block.

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

12.1 Tables
Table 1: Referenced Namespaces .......................................................... 10
Table 2: Predefined business scenario types ............................................ 16
Table 3: Defined URIs for the WS Addressing Action element .................. 19
Table 4: Digest method: allowed algorithm identifiers ............................ 22
Table 5: Signature method: allowed algorithm identifiers ......................... 22
Table 6: Symmetric encryption algorithms ............................................. 25
Table 7: Security token types – support requirements .............................. 26
Table 8: Predefined business scenario types ............................................ 48
Table 9: OSCI X.509-Token usages ...................................................... 96
Table 10: SOAP/OSCI roles assigned to token usages .............................. 97

12.2 Pictures
Figure 1: Actors and nodes involved in the message flow .......................... 12
Figure 2: Request Security Token Message ........................................... 29
Figure 2: Request Security Token, Body for Issue Request ....................... 31
Figure 3: Request Security Token Response Message ............................. 32
Figure 4: Request Security Token, Body for Issue Response ..................... 33
Figure 5: SAML 2.0 Assertion constituents ............................................ 34
Figure 6: RST for OneTimeToken .......................................................... 38
Figure 7: RSTR for OneTimeToken ....................................................... 39
Figure 8: MessageMetaData overview .................................................. 69
Figure 9: osci:Request header and body block assembly .......................... 82
Figure 10: osci:Response header and body block assembly ....................... 84
Figure 11: MsgBoxFetchRequest header and body block assembly .............. 86
Figure 12: MsgBoxStatusListRequest header and body block assembly ....... 87
Figure 13: MsgBoxResponse header and body block assembly ................... 88
Figure 14: MsgBoxGetNextRequest header and body block assembly .......... 89
Figure 15: MsgBoxClose header and body block assembly ........................ 91

12.3 OSCI specific faults
Fault 1: ProcessingException .............................................................. 13
Fault 2: AddrWrongActionURI ............................................................ 19
Fault 3: AddrWrongTypeOfBusinessScenario ........................................ 19
12.4 Listings

Listing 1: ExampleEndpointOSCIPolicy.xml ......................................................... 125
Listing 2: Example XML Signature ................................................................. 127
13 References

13.1 Normative


OSCI-Transport Version 2.0.1 – Specification

© 2013 Koordinierungsstelle für IT-Standards / Coordination Office for IT Standards

open.org/wss/v1.1/wss-v1.1-spec-errata-os-
SOAPMessageSecurity.pdf

[WSSC] Web Services Secure Conversation 1.3, OASIS Standard, 1 March 2007,
http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/ws-secureconversation-1.3-os.pdf

[WSSP] WS-SecurityPolicy 1.2, OASIS Standard 1 July 2007,
http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/ws-securitypolicy-1.2-spec-os.pdf

[WSSKERB] Web Services Security Kerberos Token Profile 1.1, OASIS Standard Specification,
incorporating Approved Errata, 1 November 2006,
http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-errata-os-
KerberosTokenProfile.pdf

[WSSSAML] Web Services Security SAML Token Profile 1.1, OASIS Standard Specification,
incorporating Approved Errata, 1 November 2006,

UsernameTokenProfile.pdf

http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-errata-os-
x509TokenProfile.pdf

[WST] WS-Trust 1.3, OASIS Standard, 19 March 2007,
http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-os.pdf


http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/


http://www.w3.org/TR/xmlsig-core/

<table>
<thead>
<tr>
<th>Reference</th>
<th>Description</th>
</tr>
</thead>
</table>

**13.2 Informative**

<table>
<thead>
<tr>
<th>Reference</th>
<th>Description</th>
</tr>
</thead>
</table>
Appendix A. Schema OSCI Transport 2.01

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:sci="http://www.osci.eu/ws/2008/05/transport"
  targetNamespace="http://www.osci.eu/ws/2013/02/transport"
  xmlns:wsa="http://www.w3.org/2005/08/addressing"
  xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
  schemaLocation="http://www.w3.org/2003/05/soap-envelope"/>

1.0.xsd" xmlns:s12="http://www.w3.org/2003/05/soap-envelope"/

<xs:complexType name="MsgTimeStampsType">
  <xs:simpleContent>
    <xs:extension base="xs:anyURI">" xmlns:sci="http://www.osci.eu/ws/2008/05/transport"
  targetNamespace="http://www.osci.eu/ws/2013/02/transport"
  xmlns:wsa="http://www.w3.org/2005/08/addressing"
  xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
  schemaLocation="http://www.w3.org/2003/05/soap-envelope"/

    <xs:element name="Initiator" type="sci:TypeOfBusinessScenarioType"/>
    <xs:element name="Date, when this message is obsolete; may be set by
    Initiator" type="xs:dateTime" minOccurs="0"/>
    <xs:element name="Delivery" type="xs:dateTime" minOccurs="0"/>
    <xs:element name="Time of entry in a Recipient's MsgBox" type="xs:dateTime" minOccurs="0"/>

    <xs:element name="InitialFetch" type="xs:dateTime" minOccurs="0"/>

© 2013 Koordinierungsstelle für IT-Standards / Coordination Office for IT Standards
<xs:annotation>
  <xs:documentation>Time of first committed fetch from MsgBox by the Recipient</xs:documentation>
</xs:annotation>
<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:complexType name="MsgBoxRequestType">
  <xs:sequence>
    <xs:element ref="osci:MsgSelector" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
<xs:simpleType name="MsgBoxReasonEnum">
  <xs:restriction base="xs:anyURI">
    <xs:enumeration value="http://www.osci.eu/ws/2008/05/transport/MsgBox/reasons/NoMatch"/>
    <xs:enumeration value="http://www.osci.eu/ws/2008/05/transport/MsgBox/reasons/SearchArgsInvalid"/>
    <xs:enumeration value="http://www.osci.eu/ws/2008/05/transport/MsgBox/reasons/RequestIdInvalid"/>
  </xs:restriction>
</xs:simpleType>
<xs:complexType name="MsgBoxReasonOpenEnum">
  <xs:union memberTypes="osci:MsgBoxReasonEnum xs:anyURI"/>
</xs:complexType>
<xs:complexType name="MsgBoxResponseType">
  <xs:choice>
    <xs:element name="NoMessageAvailable">
      <xs:complexType>
        <xs:attribute name="reason" type="osci:MsgBoxReasonOpenEnum" use="required"/>
      </xs:complexType>
    </xs:element>
    <xs:element name="ItemsPending" type="xs:nonNegativeInteger"/>
  </xs:choice>
  <xs:attribute name="MsgBoxRequestID" type="xs:anyURI" use="required"/>
</xs:complexType>
<xs:complexType name="MsgAttributeListType">
  <xs:sequence>
    <xs:element ref="wsa:MessageID" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element ref="wsa:From" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element ref="osci:TypeOfBusinessScenario"/>
    <xs:element name="MsgSize" type="xs:int"/>
    <xs:element name="ObsoleteAfterDate" type="xs:date" minOccurs="0"/>
    <xs:element name="DeliveryTime" type="xs:dateTime"/>
    <xs:element name="InitialFetchedTime" type="xs:dateTime" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="MsgBoxRequestID" type="xs:anyURI"/>
<xs:complexType name="MsgStatusListType">
  <xs:sequence>
    <xs:element name="MsgAttributes" type="osci:MsgAttributeListType" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element ref="osci21:MessageMetaData" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:element name="MsgBoxFetchRequest">
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="osci:MsgBoxRequestType">
        <xs:attribute name="MsgPart" default="Envelope">
          <xs:simpleType>
            <xs:restriction base="xs:NMTOKEN">
              <xs:enumeration value="Envelope"/>
              <xs:enumeration value="Header"/>
              <xs:enumeration value="Body"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:attribute>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>

<xs:element name="MsgBoxStatusListRequest" type="osci:MsgBoxStatusListRequestType"/>
<xs:complexType name="MsgBoxStatusListRequestType">
  <xs:complexContent>
    <xs:extension base="osci:MsgBoxRequestType">
      <xs:attribute name="maxListItems" type="xs:positiveInteger"/>
      <xs:attribute name="ListForm">
        <xs:simpleType>
          <xs:restriction base="xs:NMTOKEN">
            <xs:enumeration value="MsgAttributes"/>
            <xs:enumeration value="MessageMetaData"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:element name="MsgBoxResponse" type="osci:MsgBoxResponseType"/>
<xs:element name="MsgBoxGetNextRequest" type="osci:MsgBoxGetNextRequestType"/>

<xs:complexType name="ReceiptDemandType">
  <xs:sequence>
    <xs:element ref="wsa:ReplyTo"/>
    <xs:attribute name="qualTSPForReceipt" type="xs:boolean" default="false"/>
  </xs:sequence>
</xs:complexType>

<xs:element name="LastMsgReceived" type="wsa:AttributedURIType" maxOccurs="unbounded"/>
<xs:attribute name="MsgBoxRequestID" type="xs:anyURI" use="required"/>
<xs:attribute name="MsgBoxCloseRequestID" type="osci:MsgBoxCloseRequestType" maxOccurs="unbounded"/>
<xs:attribute name="LastMsgReceived" type="wsa:AttributedURIType" maxOccurs="unbounded"/>
<xs:attribute name="MsgBoxRequestID" type="xs:anyURI" use="required"/>
<xs:attribute name="MsgBoxCloseRequestID" type="osci:MsgBoxCloseRequestType" maxOccurs="unbounded"/>
<xs:attribute name="qualTSPForReceipt" type="xs:boolean" default="false"/>
<xs:attribute name="echoRequest" type="xs:boolean" default="false"/>
<xs:attribute name="ReceiptDemandType"/>
<xs:element ref="wsa:ReplyTo"/>
<xs:attribute name="qualTSPForReceipt" type="xs:boolean" default="false"/>

© 2013 Koordinierungsstelle für IT-Standards / Coordination Office for IT Standards
<xs:attribute name="echoRequest" type="xs:boolean" default="false"/>
</xs:complexType>

<xs:element name="DeliveryReceiptDemand" type="osci:DeliveryReceiptDemandType"/>
<xs:element name="ReceptionReceiptDemand" type="osci:ReceptionReceiptDemandType"/>
<xs:element name="ReceiptInfo" type="osci:ReceiptInfoType"/>
<xs:complexType name="ReceiptInfoType">
  <xs:sequence>
    <xs:element ref="wsa:MessageID"/>
    <xs:element ref="osci:MsgTimeStamps"/>
    <xs:element ref="wsa:RelatesTo" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element ref="wsa:From" minOccurs="0"/>
    <xs:element ref="wsa:ReplyTo"/>
    <xs:element name="RequestEcho" type="xs:base64Binary" minOccurs="0"/>
    <xs:element ref="osci21:MessageMetaData" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="Id" type="xs:ID" use="required"/>
  <xs:attribute name="ReceiptIssuerRole" use="optional">
    <xs:simpleType>
      <xs:restriction base="xs:anyURI">
        <xs:enumeration value="http://www.osci.eu/ws/2008/05/transport/role/MsgBox"/>
        <xs:enumeration value="http://www.osci.eu/ws/2008/05/transport/role/Recipient"/>
        <xs:enumeration value="http://www.osci.eu/ws/2008/05/transport/role/Sender"/>
        <xs:enumeration value="http://www.osci.eu/ws/2008/05/transport/role/Relay"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:restriction base="xs:anyURI">
      <xs:enumeration value="http://www.osci.eu/ws/2008/05/transport/role/Recipient"/>
      <xs:enumeration value="http://www.osci.eu/ws/2008/05/transport/role/Sender"/>
      <xs:enumeration value="http://www.osci.eu/ws/2008/05/transport/role/Relay"/>
    </xs:restriction>
  </xs:attribute>
</xs:complexType>

<xs:complexType name="DeliveryReceiptDemandType">
  <xs:complexContent>
    <xs:restriction base="osci:ReceiptDemandType">
      <xs:sequence>
        <xs:element ref="wsa:ReplyTo"/>
      </xs:sequence>
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="ReceptionReceiptDemandType">
  <xs:complexContent>
    <xs:restriction base="osci:ReceiptDemandType">
      <xs:sequence>
        <xs:element ref="wsa:ReplyTo"/>
      </xs:sequence>
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="FetchedNotificationDemandType">
  <xs:sequence>
    <!-- xs:attribute ref="s12:role" fixed="http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver" -->
  </xs:sequence>
</xs:complexType>

<xs:complexType name="DeliveryReceiptType">
  <xs:sequence>
    <xs:element ref="osci:ReceiptInfo"/>
    <xs:element ref="ds:Signature"/>
  </xs:sequence>
</xs:complexType>

<xs:element name="DeliveryReceipt" type="osci:DeliveryReceiptType"/>
<xs:element name="SubmissionReceipt" type="osci:DeliveryReceiptType"/>
<xs:element name="RelayReceipt" type="osci:DeliveryReceiptType"/>
<xs:complexType name="ReceptionReceiptType">
  <xs:sequence>
    <xs:element ref="osci:ReceiptInfo"/>
    <xs:element ref="ds:Signature"/>
  </xs:sequence>
</xs:complexType>

<xs:element name="ReceptionReceipt" type="osci:ReceptionReceiptType"/>
<xs:complexType name="FetchedNotificationDemandType">
  <xs:sequence>
    <!-- xs:attribute ref="s12:role" fixed="http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver" -->
  </xs:sequence>
</xs:complexType>

© 2013 Koordinierungsstelle für IT-Standards / Coordination Office for IT Standards
<xs:element ref="wsa:ReplyTo"/>
</xs:sequence>
</xs:complexType>

<xs:element name="FetchedNotificationDemand" type="osci:FetchedNotificationDemandType"/>
<xs:complexType name="FetchedNotificationType">
<xs:sequence>
<xs:element name="FetchedTime" type="xs:dateTime"/>
<xs:element ref="wsa:MessageID"/>
<xs:element ref="wsa:To"/>
<xs:element ref="wsa:From"/>
</xs:sequence>
</xs:complexType>

<xs:element name="FetchedNotification" type="osci:FetchedNotificationType"/>
<!--Extentensions for Key usage context-->
<xs:complexType name="X509TokenContainerType">
<xs:sequence maxOccurs="unbounded">
<xs:element ref="osci:X509TokenInfo"/>
</xs:sequence>
<xs:attribute name="validateCompleted" type="xs:boolean" default="false"/>  
</xs:complexType>

<xs:element name="X509TokenContainer" type="osci:X509TokenContainerType"/>
<xs:element name="X509TokenInfo">
<xs:complexType>
<xs:sequence>
<xs:element ref="ds:X509Data" maxOccurs="unbounded"/>
<xs:complexType>
<xs:element name="TimeInstant" type="xs:dateTime"/>
</xs:complexType>
</xs:sequence>
</xs:complexType>

<!-- RFC 3280 for KeyUsage with Extentensions Attribute Certificate and usage for Authentication -->
<xs:complexType>
<!--OSCI Policy Asserstions-->  
<!--Policy qualified Timestamp Servcie available-->  
<xs:element>
<!--Policy Assertion carrying Endpoints X509Certificates-->  
<xs:element name="X509CertificateAssertion">
<xs:complexType>
<xs:sequence>
<xs:element ref="wsp:All"/>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:complexType>

<!--Policy, when qualified TSP service can be requested from this node-->  
<xs:element name="QualTspAssertion">
<xs:complexType>
<xs:attribute name="PolicyRef" type="xs:anyURI"/>
</xs:complexType>
</xs:element>

<!--Policy if and how MsgTimeStamps:ObsoleteAfter is handled-->  
<xs:element name="ObsoleteAfterAssertion">
<xs:complexType>
<xs:attribute name="PolicyRef" type="xs:anyURI"/>
</xs:complexType>
</xs:element>

<xs:element name="MsgRetainDays" type="xs:positiveInteger" minOccurs="0"/>
<xs:element name="WarningBeforeMsgObsolete" type="xs:positiveInteger" minOccurs="0"/>
</xs:sequence>
</xs:complexType>

<!--Policy if and how Message is handled-->  
<xs:element name="ObsoleteAfterAssertion">
<xs:complexType>
<xs:attribute name="PolicyRef" type="xs:anyURI"/>
</xs:complexType>
</xs:element>

<xs:element name="WarningBeforeMsgObsolete" type="xs:positiveInteger" minOccurs="0"/>  
</xs:sequence>
</xs:complexType>

<xs:attribute name="PolicyRef" type="xs:anyURI"/>  
</xs:element>
4182   </xs:complexType>
4183 </xs:element>
4184 <!--Policy for MakeConnection: Response Retention Days-->
4185 <xs:element name="MsgRetainDays" type="xs:positiveInteger"/>
4186 <!--Enumerate for possible X509 Token Usages-->
4187 <xs:attribute name="TokenUsage">
4188   <xs:simpleType>
4189     <xs:restriction base="xs:anyURI">
4190       <xs:enumeration
4191         value="http://www.osci.eu/common/names/TokenUsage/e2eContentEncryption"/>
4192       <xs:enumeration
4193         value="http://www.osci.eu/common/names/TokenUsage/TransportEncryption"/>
4194       <xs:enumeration
4195         value="http://www.osci.eu/common/names/TokenUsage/ReceiptSigning"/>
4196       <xs:enumeration
4197         value="http://www.osci.eu/common/names/TokenUsage/TSPSigning"/>
4198     </xs:restriction>
4199   </xs:simpleType>
4200 </xs:attribute>
4201 <!--Opaque Body Type - not used-->
4202 <!--Policy maximum accepted Message size and Frequency per hour-->
4203 <xs:element name="AcceptedMsgLimits">
4204   <xs:complexType>
4205     <xs:sequence>
4206       <xs:element name="MaxSize" type="xs:positiveInteger"/>
4207       <xs:element name="MaxPerHour" type="xs:positiveInteger"/>
4208     </xs:sequence>
4209   </xs:complexType>
4210 </xs:element>
4211 <xs:complexType name="MessageBody">
4212   <xs:sequence>
4213     <xs:any namespace="##any" minOccurs="0" maxOccurs="unbounded"/>
4214   </xs:sequence>
4215 </xs:complexType>
4216 </xs:schema>
Appendix B. OSCI Transport 2.01 – Schema MessageMetaData

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xsi="http://www.w3.org/2001/XMLSchema"
xmlns:secext="http://www.osci.eu/ws/2013/02/transport"
targetNamespace="http://www.osci.eu/ws/2013/02/transport"
elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:import namespace="http://www.w3.org/2005/08/addressing" schemaLocation="ws-addr.xsd"/>
  <xs:import namespace="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" schemaLocation="oasis-200401-wss-wssecurity-secext-1.0.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>
  <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">
        <xs:annotation>
          <xs:documentation>Destination node: Successful delivery to recipient in synchronous scenarios, to MsgBox if asynchronous</xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="Fetch" minOccurs="0">
        <xs:annotation>
          <xs:documentation>Only MsgBox node: Initial fetch of message by recipient from his MsgBox</xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="Reception" minOccurs="0">
        <xs:annotation>
          <xs:documentation>Ultimate Recipient node, after acceptance of message, after successful decryption of payload</xs:documentation>
        </xs:annotation>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

© 2013 Koordinierungsstelle für IT-Standards / Coordination Office for IT Standards
<xs:annotation>
  <xs:element name="ReceiptTo" type="wsa:EndpointReferenceType" minOccurs="0"/>
</xs:annotation>

<xs:complexType name="DeliveryAttributesType">
  <xs:documentation>Message delivery time instants, quality and receipts requested</xs:documentation>
</xs:complexType>

<xs:sequence>
  <xs:documentation>Timestamps, priority etc.</xs:documentation>
</xs:sequence>

<xs:element name="Origin" type="xs:dateTime" minOccurs="0"/>
</xs:element>

<xs:documentation>Prodction of content by requester respective (response) provider</xs:documentation>

<xs:element name="InitialSend" type="xs:dateTime" minOccurs="0"/>
</xs:element>

<xs:documentation>Time when delivery was started (submission by senders node)</xs:documentation>

<xs:element name="NotBefore" type="xs:dateTime" minOccurs="0"/>
</xs:element>

<xs:documentation>Time when sending node should submit message</xs:documentation>

<xs:element name="ObsoleteAfter" type="xs:date" minOccurs="0"/>
</xs:element>

<xs:documentation>Date, when this message is obsolete; may be set by initiator</xs:documentation>

<xs:element name="Delivery" type="xs:dateTime" minOccurs="0"/>
</xs:element>

<xs:documentation>Time of entry in a recipients MsgBox or reception by recipient in synchronous case</xs:documentation>

<xs:element name="InitialFetch" type="xs:dateTime" minOccurs="0"/>
</xs:element>

<xs:documentation>Time of first comitted fetch from MsgBox by recipient</xs:documentation>

<xs:element name="Reception" type="xs:dateTime" minOccurs="0"/>
</xs:element>

<xs:documentation>Reception time set by the Ultimate Recipient ("Reader", target application)</xs:documentation>

<xs:element name="ServiceQuality" minOccurs="0"/>
</xs:element>

<xs:documentation>Property like priority etc. - to be detailed</xs:documentation>

<xs:complexType>
  <xs:complexContent>
    <xs:extension base="osci21:PropertyType"/>
  </xs:complexContent>
</xs:complexType>

<xs:element name="ReceiptRequests" type="osci21:ReceiptRequestType" minOccurs="0"/>
</xs:element>
Receipts requested by sender or author

Logical identifier and optional authentication token (binary, may carry SAML, too)

Value of generic party identifier, as classified by @type 

Value of "friendly name" value for displaying in user agents (as e.g. known from eMail)

Concrete role of party in business scenario (e.g. "buyer", "Meldesbürde", "Standesamt"...)

If response expected different from value outlined in

"From" address
<xs:complexType>
  <xs:complexContent>
    <xs:extension base="osci21:NonEmptyStringType">
      <xs:attribute name="ProcessName" type="osci21:NonEmptyStringType"/>
      <xs:documentation>Process may have a name, e.g. "order"</xs:documentation>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="ProcessIdentifierType">
  <xs:annotation>
    <xs:documentation>Process ID message is realated to</xs:documentation>
  </xs:annotation>
  <xs:simpleContent>
    <xs:extension base="osci21:NonEmptyStringType">
      <xs:attribute name="ProcessRef" type="osci21:NonEmptyStringType" minOccurs="0"/>
      <xs:documentation>References to business process ID's (like ebMS Conversation ID, "Aktenzeichen" in Germany)</xs:documentation>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="MsgIdentificationType">
  <xs:sequence>
    <xs:element ref="wsa:MessageID"/>
    <xs:element name="In-Reply-To" type="wsa:AttributedURIType" minOccurs="0" maxOccurs="unbounded"/>
    <xs:documentation>Referenced application level Message ID(s)</xs:documentation>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="PropertyType">
  <xs:sequence>
    <xs:element name="Requester" type="osci21:ProcessIdentifierType" minOccurs="0"/>
    <xs:element name="Responder" type="osci21:ProcessIdentifierType" minOccurs="0"/>
    <xs:documentation>Ref on requester (Source Application) side</xs:documentation>
    <xs:element name="Responder" type="osci21:ProcessIdentifierType" minOccurs="0"/>
    <xs:documentation>Ref on responder (Target Application) side, if different</xs:documentation>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="PropertyType">
  <xs:sequence>
    <xs:element name="Name" type="xs:QName" use="required"/>
    <xs:documentation>Name of property (in scheme denoted)</xs:documentation>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="MessagePropertiesType">
  <xs:sequence>
    <xs:element name="Property" type="xs:documentation"/>
  </xs:sequence>
</xs:complexType>
<xs:element name="Property" type="osci21:PropertyType" maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>

<xs:complexType name="QualifierType">
  <xs:sequence>
    <xs:element name="Subject" type="xs:string" minOccurs="0"/>
  </xs:annotation>
  <xs:documentation>Message subject text (informational)</xs:documentation>
</xs:element>
</xs:complexType>

<xs:element name="BusinessScenario" type="xs:QName">
  <xs:annotation>
  </xs:documentation>Related business scenario. QName must be defined in a namespace</xs:documentation>
</xs:element>

<xs:element name="Service" type="xs:anyURI">
  <xs:annotation>
  </xs:documentation>Distinct service in a certain business scenario context; in the XÖV context this is the "Dienste URI"
</xs:documentation>
</xs:element>

<xs:element name="MessageID">
  <xs:annotation>
  </xs:documentation>Message ID and originators and reply address</xs:documentation>
</xs:element>

<xs:element name="DeliveryAttributes" type="osci21:DeliveryAttributesType">
  <xs:annotation>
  </xs:documentation>Timestamps, receipts to be generated, service quality</xs:documentation>
</xs:element>

<xs:element name="Originators" type="osci21:OriginatorsType">
  <xs:annotation>
  </xs:documentation>Message originators and reply address</xs:documentation>
</xs:element>

<xs:element name="Destinations" type="osci21:DestinationsType">
  <xs:annotation>
  </xs:documentation>Actual and other destinations of message</xs:documentation>
</xs:element>

<xs:element name="MsgIdentification" type="osci21:MsgIdentificationType">
  <xs:annotation>
  </xs:documentation>Message ID and message relations</xs:documentation>
</xs:element>

<xs:element name="Qualifier" type="osci21:QualifierType">
  <xs:annotation>
  </xs:documentation>General payload properties, common to all scenarios</xs:documentation>
</xs:element>

<xs:element name="MessageProperties">
  <xs:annotation>
  </xs:documentation>Scenarios specific payload properties, to be agreed upon per scenario</xs:documentation>
</xs:element>

© 2013 Koordinierungsstelle für IT-Standards / Coordination Office for IT Standards
<xs:complexType>
  <xs:sequence>
    <xs:element name="Property" type="osci21:PropertyType" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<xsl:element name="Sender" type="osci21:PartyType">
  <xs:annotation>
    <xs:documentation>Sending node, entry may be added by Sender node</xs:documentation>
  </xs:annotation>
</xsl:element>

<xsl:element name="OtherDestinations">
  <xs:annotation>
    <xs:documentation>Other destinations of message - informational, as known from e-mail</xs:documentation>
  </xs:annotation>
</xsl:element>

<xsl:element name="OtherReaders" type="osci21:PartyIdentifierType"/>

<xsl:element name="CcReaders" type="osci21:PartyIdentifierType">
  <xs:annotation>
    <xs:documentation>Destinations in cc role</xs:documentation>
  </xs:annotation>
</xsl:element>

<xsl:element name="MessageMetaData">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="osci21:DeliveryAttributes"/>
      <xs:element ref="osci21:Originators"/>
      <xs:element ref="osci21:Destinations"/>
      <xs:element ref="osci21:MsgIdentification"/>
      <xs:element ref="osci21:Qualifier"/>
      <xs:element ref="osci21:MessageProperties" minOccurs="0"/>
      <xs:element name="MsgSize" type="xs:positiveInteger" minOccurs="0">
        <xs:annotation>
          <xs:documentation>Message size in bytes</xs:documentation>
        </xs:annotation>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xsl:element>

<xsl:attribute name="TestMsg" type="xs:boolean" default="false">
  <xs:annotation>
    <xs:documentation>"true", if test-message; defaults to "false"</xs:documentation>
  </xs:annotation>
</xsl:attribute>

</xs:schema>
Appendix C. Example: OSCI Endpoint Metadata Instance

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

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

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

```xml
<?xml version="1.0" encoding="UTF-8"?>
<wsp:Policy wsu:id="X509CertificateAssertion">
  <osci:X509CertificateAssertion>
    <wsp:All>
        <wss:Reference URI="REPLACE_WITH_ACTUAL_URL to UltimateRecipientEncCert" ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"/>
      </wss:SecurityTokenReference>
        <wss:Reference URI="REPLACE_WITH_ACTUAL_URL to RecipientSigCert" ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"/>
      </wss:SecurityTokenReference>
      <wss:SecurityTokenReference wsu:id="MsgBoxEncCert" wsse:Usage="http://www.osci.eu/2008/05/common/names/TokenUsage/TransportEncryption" osci:Role="http://www.osci.eu/2008/05/common/names/role/MsgBox">
      </wss:SecurityTokenReference>
    </wsp:All>
  </osci:X509CertificateAssertion>
</wsp:Policy>
```
Listing 1: ExampleEndpointOSCIPolicy.xml
Appendix D. Example Signature Element

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

```
<ds:Signature Id="uuid:E57C0006-A629-5767-ED32-2667F1512912">
  <ds:SignedInfo>
    <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
    <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/>
    <ds:Reference URI="#uuid:97544A28-F042-9457-3286-DD37F6FF7FEA">
      <ds:Transforms>
        <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
      </ds:Transforms>
      <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
      <ds:DigestValue>DQrljZZVeewWoXLzLLi/uPqESY2fGscAjVlBxjKEEnM=</ds:DigestValue>
    </ds:Reference>
      <ds:Transforms>
        <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
      </ds:Transforms>
      <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
      <ds:DigestValue>RjwjBl2TBumKSvG05Jgelzz6jll1A3t7GUXikLaa8io=</ds:DigestValue>
    </ds:Reference>
  </ds:SignedInfo>
  <ds:SignatureValue>FrPlHt0v/Njn4j84T8vO/VnyE...8JZV/LE141aSTcLyBx8Q==</ds:SignatureValue>
  <ds:KeyInfo>
    <ds:X509Data>
      <ds:X509Certificate>MIIDHjCCAgagAwIBAAIER4...YQya8Q==</ds:X509Certificate>
    </ds:X509Data>
  </ds:KeyInfo>
  <ds:Object>
    <xades:QualifyingProperties Target="#uuid:E57C0006-A629-5767-ED32-2667F1512912">
      <xades:SignedProperties>
        <xades:SignedSignatureProperties Id="uuid:5A075139-52EB-CF5E-3A1B-F54B6B1F1025">
          <xades:SigningTime>2008-01-17T18:57:27</xades:SigningTime>
          <xades:SigningCertificate>
            <xades:Cert>
              <xades:CertDigest>
                <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
                <ds:DigestValue>OuGTT8Gg...oXRjpsKp9BMVBaYid7kza4=</ds:DigestValue>
              </xades:CertDigest>
            </xades:Cert>
          </xades:SigningCertificate>
        </xades:SignedSignatureProperties>
      </xades:SignedProperties>
    </xades:QualifyingProperties>
  </ds:Object>
</ds:Signature>
```
Listing 2: Example XML Signature

```xml
<xades:Signature>
  <xades:CertiDigest>
  </xades:CertiDigest>
  <xades:IssuerSerial>
    <ds:X509IssuerName>CN=Apitzsch, OU=QA, O=waycony, L=Hamburg, C=DE</ds:X509IssuerName>
    <ds:X509SerialNumber>1200577645</ds:X509SerialNumber>
  </xades:IssuerSerial>
  </xades:Cert>
  </xades:SigningCertificate>
  </xades:SignedSignatureProperties>
  </xades:SignedProperties>
  <xades:UnsignedProperties>
    <xades:UnsignedSignatureProperties>
      <xades:SignatureTimeStamp>
        <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
        <xades:EncapsulatedTimeStamp>0uGKTBGg..oXRjpsKp9BMVBaYid</xades:EncapsulatedTimeStamp>
      </xades:SignatureTimeStamp>
    </xades:UnsignedSignatureProperties>
  </xades:UnsignedProperties>
  </xades:QualifyingProperties>
</ds:Object>
</ds:Signature>
```
## Appendix E. Change History

<table>
<thead>
<tr>
<th>Version as of</th>
<th>Author</th>
<th>Changes made in chapter / Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.0.1 04/07/13</td>
<td>A. Wall, J. Apitzsch</td>
<td>Finalization, formal QA</td>
</tr>
</tbody>
</table>
Appendix F. Acknowledgements

This revision of the OSCI 2.0 Transport specification first of all profited from work done in the years 2012-2013 by the technical working group of the project "XTA" of the German "IT-Planungs rat".