



OIO Reliable Asynchronous Profile version 1.2



IT- og Telestyrelsen
Ministeriet for Videnskab
Teknologi og Udvikling



OIO Reliable Asynchronous Secure
Profile version 1.2

OIO RASP 1.2

Published by:
The National IT and Telecom Agency

The National IT and
Telecom Agency
Holsteinsgade 63
DK-2100 Copenhagen Ø
Denmark

Tel: 3545 0000

Fax: 3545 0010

>

OIO Reliable Asynchronous Secure Profile

Version 1.2

OIO RASP 1.2

Contents

>

Introduction	6
About OIO RASP	6
Open process	6
Layers in the OIO Web Service Stack	6
Business requirements	7
Authentication	7
Confidentiality	7
Integrity	7
Reliability	7
Generation of evidence	7
Intermediaries	7
Asynchronous communication	7
Document Conventions	8
Notational Conventions	8
Profile Identification and Versioning	8
OIO Basic Profile 1.1	9
SOAP 1.2 Envelopes	9
Empty SOAP Bodies	9
SMTP Binding	9
OIO Basic Security Profile 1.1	11
Encryption of messages	11
Signing before encryption	11
Including certificates in messages	12
Cryptographic Algorithms	12
OIO Reliable Messaging 1.1	14
Exponential backoff	14
Acknowledgements	14
OIO RASP SOAP Headers	15
Namespace	15
mustUnderstand property	15
Custom header blocks	15
Schema	17
Example (non-normative)	18
OIO RASP SOAP Faults	20
Rules for Code	20
Rules for Reason	20
WS-Security and Custom SOAP Faults	20
OIO RASP SOAP Faults	21
Example Fault	22
OIO RASP Ports	23
References	24
Appendix A - Requirements Checklist	25

Appendix B - OIO RASP Reliable Messaging Properties	29
Appendix C - Sequence Diagram	30
Appendix D - OIO RASP usage in NemHandel	31
Introduction	31
About NemHandel	31
Sending OIO business documents	31
SOAP actions to use in RASP calls	31

Introduction

>

This document defines the OIO Reliable Asynchronous Secure Profile version 1.2 (hereafter, "Profile" or OIO RASP), consisting of a set of non-proprietary Web services specifications, along with clarifications, refinements, interpretations and amplifications of those specifications which promote interoperability.

About OIO RASP

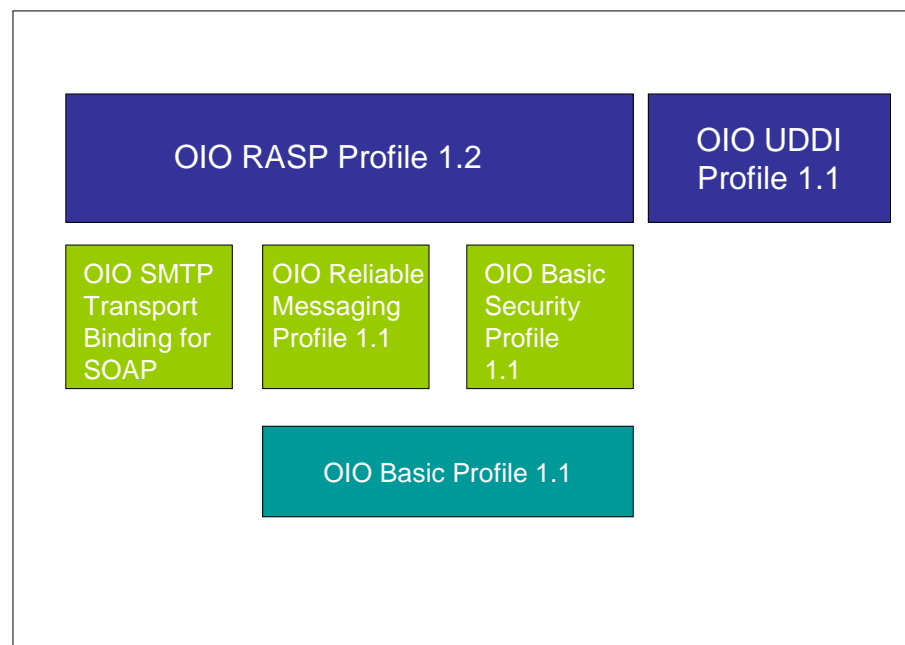
OIO Reliable Asynchronous Secure Profile (OIO RASP) is a profile of web service standards from the WS-* family of standards. It supports asynchronous exchange of business documents via the Internet with a high degree of security and reliability.

Open process

OIO RASP has been developed in an open and accountable process at the Danish National IT and Telecom Agency. Input has been given from several public sector institutions as well as private companies through a series of workshops in 2006.

Layers in the OIO Web Service Stack

The figure below illustrates the different layers in the OIO web service stack and their mutual relations:



Note: The OIO UDDI profile is not part of the RASP profile but is composable with RASP.

Business requirements

>

This profile has been developed with the following set of business requirements in mind:

Authentication

It must be possible to authenticate the sender of a business document.

Confidentiality

Only the sender and the receiver may know the content of the business document.

Integrity

It must not be possible for a third party to tamper with business documents during transit from sender to receiver.

Reliability

Documents should not be lost in transit even if the underlying communication infrastructure is unreliable.

Generation of evidence

It should be possible to generate digital evidence that can be presented in a court of law. The receiver of a business document should have evidence that the sender did send the document, and the sender of a document should have evidence that the receiver did receive the document.

Intermediaries

It should be possible to send a business document between two parties via an intermediary party.

Asynchronous communication

Asynchronous communication should be possible using communication channels available to SMEs such as HTTP and SMTP.

Document Conventions

>

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.

Normative statements of requirements in the Reliable Asynchronous Secure Profile 1.2 (i.e., those impacting conformance, as outlined in "Conformance Requirements") are presented in the following manner:

> RSP-nnnn *Statement text here.*

where "nnnn" is replaced by a number that is unique among the requirements in the Reliable Asynchronous Secure Profile 1.2, thereby forming a unique requirement identifier.

Profile Identification and Versioning

This document is identified by a name and a version number. Together, they identify a particular profile instance.

Version numbers are composed of a major and minor portion, in the form "major.minor". They can be used to determine the precedence of a profile instance; a higher version number (considering both the major and minor components) indicates that an instance is more recent, and therefore supersedes earlier instances.

Instances of profiles with the same name (e.g., "Example Profile 1.1" and "Example Profile 5.0") address interoperability problems in the same general scope (although some developments may require the exact scope of a profile to change between instances).

One can also use this information to determine whether two instances of a profile are backwards-compatible; that is, whether one can assume that conformance to an earlier profile instance implies conformance to a later one. Profile instances with the same name and major version number (e.g., "Example Profile 1.1" and "Example Profile 1.2") MAY be considered compatible. Note that this does not imply anything about compatibility in the other direction; that is, one cannot assume that conformance with a later profile instance implies conformance to an earlier one.

OIO Basic Profile 1.1

>

This section incorporates the OIO Basic Profile 1.1 by reference. The OIO Basic Profile 1.1 (OIO BP 1.1) is a Danish localization and customization of the WS-I Basic Profile 1.1 with errata.

This profile (OIO RASP 1.2) however defines a number of overrides and extensions to OIO BP 1.1.



RSP-1001: OIO RASP services **MUST** conform to the rules of the OIO BP 1.1 except for the overrides and extensions specified in OIO RASP 1.2.

SOAP 1.2 Envelopes

Interoperability testing of different versions of SOAP in combination with namely WS-Security and WS-ReliableMessaging proved to be less problematic when SOAP 1.2 was used. SOAP 1.2 furthermore clarifies a number of issues raised with SOAP 1.1.

SOAP 1.2 has therefore been chosen as standard for envelopes over SOAP 1.1.



RSP-1002: An Envelope **MUST** conform to the structure specified in SOAP 1.2 Section 5, "SOAP Message Construct". SOAP 1.1 envelopes **MUST NOT** be used.

Empty SOAP Bodies

Response messages that do not contain SOAP faults should be empty:



RSP-1003:
The SOAP body of the response message of the application level web service request/response that does not contain a SOAP fault **SHOULD** be empty.

The receiver of a SOAP body of the response message of the application level web service request/response that does not contain a SOAP fault **SHOULD** ignore any possible contents of the SOAP body.

SMTP Binding

Communication of business transactions must be possible using communication protocols available to small and medium sized private enterprises as well as small and medium sized public organisations (hereafter, "SMEs"). Commonly used Internet

>

protocols are HTTP and SMTP. SMTP has advantages to SMEs since the protocol and the standard infrastructure provided by Internet Solution Providers (ISPs) makes it easy to communicate two-ways in an asynchronous fashion. A SOAP binding to SMTP has therefore been allowed in OIO RASP.



RSP-1004: A Message MAY be sent using SMTP in accordance with the OIO SOAP SMTP binding.

OIO Basic Security Profile 1.1

>

This section of the Reliable Asynchronous Secure Profile 1.1 incorporates the OIO Basic Security Profile 1.1 (OIO BSP 1.1) by reference. The OIO Basic Security Profile 1.1 is a Danish localization and customization of the WS-I Basic Security Profile.

This profile (OIO RASP 1.2) however defines a number of restrictions to OIO BSP 1.1.



RSP-1005: OIO RASP conformant services **MUST** conform to the rules of the OIO Basic Security Profile 1.1.

Encryption of messages

The OIO Basic Security Profile specifies how encryption can be leveraged to achieve message confidentiality.

Encryption and decryption causes an overhead to the message handling. Therefore the OIO BSP 1.1 states that encryption should be limited to confidential data. This profile mandates encryption of all payload data despite this recommendation in order to allow the infrastructure to be payload agnostic. This business requirement prohibits selective encryption based on knowledge about the payload.



RSP-1006: All messages exchanged via the OIO RASP profile **MUST** be encrypted according to the rules defined in OIO Basic Security Profile 1.1. The entire payload of the message (SOAP body) **MUST** be encrypted.

Signing before encryption

In order to ensure message authentication and integrity, all OIO RASP messages must be signed. The signing process must happen before encryption to ensure that the signer commits to plaintext data.



RSP-1007: All messages exchanged via the OIO RASP profile **MUST** be signed according to the rules defined in the OIO Basic Security Profile. Signing **MUST** happen before encryption and **MUST** include OIO RASP custom SOAP headers, OIO Reliable Messaging SOAP headers, WS-Addressing headers and the SOAP body.

User-defined (non-RASP) headers are not required to be signed.

Unsigned messages or headers **MAY** be ignored by the receiver.



RSP-1008: The signature **MUST** be made with a private key bound to an OCES Company certificate or OCES Function certificate. The latter is recommended.

The receiver of signed OIO RASP messages (including acknowledgements) should consider how digital evidence is handled. It is outside the scope of the OIO RASP profile to provide recommendations in this area and requirements may further vary considerably with the business context. IT & Telestyrelsen has published a guide on how to assure the evidence value of signatures (available from <http://www.itst.dk>) which should be consulted.

Including certificates in messages

The OIO Basic Security Profile (BSP) deals with the security of a single message sent between two parties. It recommends that a message signer either includes his certificate as a security token or references the certificate.

Since RASP deals with sequences of potentially many messages, the following additional restrictions are defined to ensure interoperability and efficiency:



RSP-1009: The sender of a message **MUST** include the sender certificate using an embedded security token as described in OIO BSP.

The receiver of a message **MUST** in replies reference the certificates of the sender and receiver using a security token reference as described OIO BSP.

The following restrictions apply to security token references:



RSP-1010: A SecurityTokenReference element **MUST** have a KeyIdentifier with the value of "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509SubjectKeyIdentifier", and an EncodingType of "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary".

Cryptographic Algorithms

To promote interoperability and security, only a small set of cryptographic algorithms are allowed in RASP:

>

RSP-1011: The following WS-Security algorithm identifiers must be used:



- Symmetric keys must be encrypted using the algorithm identified by "http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p".
- Symmetric keys must be digested using the algorithm identified by "http://www.w3.org/2000/09/xmldsig#sha1".
- The signature must use the canonicalization algorithm identified by "http://www.w3.org/2001/10/xml-exc-c14n#".
- The signature must be created using the method identified by "http://www.w3.org/2000/09/xmldsig#rsa-sha1".
- The body encryption must use the algorithm identified by the WS-Security identifier "http://www.w3.org/2001/04/xmlenc#aes256-cbc".

OIO Reliable Messaging 1.1



This section incorporates the OIO Reliable Messaging Profile 1.1 (OIO RM 1.1) by reference. The OIO RM 1.1 is a Danish customization of the WS-ReliableMessaging specification.



RSP-1012: OIO RASP messages **MUST** be sent reliably according to the rules of OIO RM 1.1. The RM protocol **SHOULD** be used in “exactly once” mode with delivery of messages in order.

Exponential backoff

The use of “exponential backoff” should be enabled whenever possible to ensure that the first retries can be made within a reasonable time limit and at the same time avoid heavy network traffic using HTTP and sending an overwhelming number of mail messages when using SMTP.



RSP-1013: Exponential backoff **SHOULD** be enabled when possible.

Acknowledgements

In order to allow the OIO Service Oriented Infrastructure the same level of assurance of delivery as with a recommended letter (using paper-based mail), the OIO RASP profile requires the receiver to sign the Reliable Messaging acknowledgements.



RSP-1014: Reliable Messaging acknowledgement messages **MUST** be signed according to the rules defined in OIO Basic Security Profile 1.1.

OIO RASP SOAP Headers



This section describes a number of custom SOAP headers to be used with OIO RASP. These among other purposes facilitate routing by intermediaries.

Namespace

The namespace property of the predefined custom OIO RASP SOAP header blocks is defined by the following:



RSP-1015: The namespace custom OIO RASP SOAP header blocks is "http://rep.oio.dk/oiosi.ehandel.gov.dk/xml/schemas/2007/09/01/"

Additional custom headers not defined in this document MAY take any namespace.

mustUnderstand property

An OIO RASP client should always indicate whether the custom header blocks are required to be understood by the receiver:



RSP-1016: Custom OIO RASP SOAP header blocks SHOULD have a mustUnderstand property which MAY be set to "true".

In cases where a mustUnderstand property is missing, or set to false, the header block MAY be ignored or handled, whichever way the recipient of the SOAP message prefers.

Custom header blocks

The table below defines the mandatory OIO RASP headers. The column "Transport/payload" determines if the header SHOULD be applied to payload messages only (i.e. the SOAP message with the XML message itself), or to all messages including transport messages (both ways, client and server).



RSP-1017: OIO RASP messages must include SenderPartyIdentifier, ReceiverPartyIdentifier and MessageIdentifier header blocks.

OIO RASP messages SHOULD include SenderPartyIdentifierType and ReceiverPartyIdentifierType header blocks.

Message Identifier SHOULD be formatted as a GUID or as the UUID (as defined in OIOUBL) in the document.

>

Element name	Description	Example	Format	Transport/Payload	Mandatory
SenderParty Identifier	<p>In contrast to the WS-Addressing “From” element, this element holds the <i>logical</i> address of the sender, e.g. an EAN number.</p> <p>In the case that no logical sender identifier exists, the element MUST be set to the value “<i>http://rep.oio.dk/oiosi.ehandel.gov.dk/xml/schemas/2007/09/01/anonymous</i>”</p> <p>When this value is set, routing should rely on the physical WS-Addressing “From” or “ReplyTo” field.</p> <p>When the receiver sends response messages (i.e. all transport messages), he must add both the “SenderPartyIdentifier” and “ReceiverParty” header and set them to the value of the received “ReceiverPartyIdentifier” and “SenderPartyIdentifier” headers, respectively.</p>	E.g. an EAN number	Non-empty string	Transport + Payload	Yes

ReceiverParty Identifier	<p>Corresponds to the EBMS “To” element. In contrast to the WS-Addressing “To” element, this element holds the <i>logical</i> address of the receiver, e.g. and EAN number.</p> <p>In the case that no logical receiver identifier exists, the element MUST set to the value “<i>http://rep.oio.dk/oiosi.ehandel.gov.dk/xml/schemas/2007/09/01/anonymous</i>”</p> <p>When this value is set, routing should rely on the physical WS-Addressing “To” field.</p> <p>When the receiver sends response messages (i.e. all transport messages), he must add both the “SenderPartyIdentifier” and “ReceiverParty” header and set them to the value of the received “ReceiverPartyIdentifier” and “SenderPartyIdentifier” headers, respectively.</p>	E.g. an EAN number	Non-empty string	Transport + Payload	Yes
SenderParty IdentifierType		“EAN”	Non-empty string	Transport + Payload	No
ReceiverParty IdentifierType		“EAN”	Non-empty string	Transport + Payload	No
Message Identifier	<p>Corresponds to the EBSM “MessageId” element. Represents a business-level message identifier, whereas the WS-Addressing message ID represents a transport-level message ID.</p>		Non-empty string	Payload	Yes

Schema

The schema below defines the syntax of the mandatory header blocks:

```
<?xml version="1.0" encoding="utf-8" ?>
<xs:schema targetNamespace=
  "http://rep.oio.dk/oiosi.ehandel.gov.dk/xml/schemas/2007/09/01/"
  elementFormDefault="qualified"
  xmlns="http://rep.oio.dk/oiosi.ehandel.gov.dk/xml/schemas/2007/09/01/"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:annotation>
    <xs:documentation>
      This Schema covers the 5 optional UBL-specific custom headers
      'VersionIdentifier', 'CustomizationIdentifier', 'ProcessIdentifier',
      'SoapBodyElementName' and 'SoapBodyElementNamespace', and the 3
      required RASP headers 'ReceiverPartyIdentifier',
      'SenderPartyIdentifier'
      and 'MessageIdentifier'. NOTE: for ISB registration, this schema should
      be split into 3 individual schemas.
    </xs:documentation>
  </xs:annotation>
</xs:schema>
```

>

```
<!-- Required RASP headers -->
<xs:element name="ReceiverPartyIdentifier" type=" NonEmptyStringType " />
<xs:element name="ReceiverPartyIdentifierType" type=" NonEmptyStringType "
minOccurs="0" />
<xs:element name="SenderPartyIdentifier" type=" NonEmptyStringType " />
<xs:element name="SenderPartyIdentifierType" type=" NonEmptyStringType "
minOccurs="0" />
<xs:element name="MessageIdentifier" type="NonEmptyStringType" />

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

</xs:schema>
```

Example (non-normative)

The example below illustrates a SOAP message with custom OIO RASP SOAP header blocks:

```
<s:Envelope xmlns:s=http://www.w3.org/2003/05/soap-envelope
  xmlns:r=http://schemas.xmlsoap.org/ws/2005/02/rm
  xmlns:a=http://www.w3.org/2005/08/addressing
  xmlns:u=http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd>
  <s:Header>
    <SenderPartyIdentifier
  xmlns=http://rep.oio.dk/oiosi.ehandel.gov.dk/xml/schemas/207/09/01>5700000000
  001</SenderPartyIdentifier>
    <SenderPartyIdentifierType
  xmlns=http://rep.oio.dk/oiosi.ehandel.gov.dk/xml/schemas/207/09/01>EAN</SenderPartyIdentifierType>
    <ReceiverPartyIdentifier
  xmlns=http://rep.oio.dk/oiosi.ehandel.gov.dk/xml/schemas/207/09/01>5700000000
  002</ReceiverPartyIdentifier>
    <ReceiverPartyIdentifierType
  xmlns=http://rep.oio.dk/oiosi.ehandel.gov.dk/xml/schemas/207/09/01>EAN</ReceiverPartyIdentifierType>
    <MessageIdentifier
  xmlns=http://rep.oio.dk/oiosi.ehandel.gov.dk/xml/schemas/207/09/01>e21a-3bb4-
  6732-3238e676-d781</MessageIdentifier>
    <!--
    ... more header blocks, such as addressing and security (signing the
    custom block)...
    -->

    <Signature xmlns=http://www.w3.org/2000/09/xmldsig#>
      <SignedInfo>
        <CanonicalizationMethod
          Algorithm=http://www.w3.org/2001/10/xml-exc-c14n# />
        <SignatureMethod
          Algorithm=http://www.w3.org/2000/09/xmldsig#rsa-sha1 />
        <Reference URI="#_8">
          <Transforms>
            <Transform Algorithm=http://www.w3.org/2001/10/xml-exc-c14n# />
          </Transforms>
          <DigestMethod Algorithm=http://www.w3.org/2000/09/xmldsig#sha1 />
          <DigestValue>vGfW6Ze3UyKMjzREe1oMk/CEz5c</DigestValue>
        </Reference>
      <!--
      ... more signature blocks...
      -->
    </SignedInfo>
```

>

```
</Signature>  
</s:Header>  
<s:Body u:Id="_1" />  
</s:Envelope>
```

OIO RASP SOAP Faults



Below a number of SOAP faults are defined for OIO RASP to ensure consistent syntax and semantics in error situations.

According to the SOAP specification, fault messages **MUST** have a Code and a Reason, and **CAN** have a Node, a Role and Detail.

Only Code and Reason are relevant for OIO RASP custom faults. Due to security issues related to exposing the internal structure of web services, they **SHOULD NOT** define a Node or a Role.



RSP-1018: OIO RASP SOAP faults **MUST** define a Code and a Reason. They **SHOULD NOT** define a Node or a Role but **CAN** define a Detail element, if detailed error messages in XML formatting are needed.

Rules for Code



RSP-1019: The Code element of a OIO RASP fault **MUST** have a top level code that is either Sender - if the fault is due to the sender of the faulted message - or Receiver - if the fault is due to an error at the receiving side.

Furthermore a OIO RASP fault **MUST** have at least one sub-code, describing what type of fault it is. If the fault was send because of an exception thrown the first level of sub-code **SHOULD** correlate to the name of the exception type.

Rules for Reason



RSP-1020: The Reason **SHOULD** contain text explaining why the fault occurred, which **SHOULD** include the error messages found in any type of exception object that might have caused the fault to be sent. Reason **MUST NOT** include stack traces from such exceptions or from elsewhere, due to security issues related to exposing the internal structure of web services.

WS-Security and Custom SOAP Faults



RSP-1021: OIO RASP faults returned **MUST** be encrypted and signed in the same way regular SOAP messages would, whenever possible.

If, however, the fault originates from a place in the communications

stack where signing and encryption is not possible (for example at transport level, where no credentials are available) the unsigned, unencrypted fault should still be sent.

OIO RASP SOAP Faults

Below is defined the custom fault codes with sub codes:

RSP-1022: The following fault codes MUST be used:

- Sender\SchemaValidationFault
- Sender\SchematronValidationFault
- Sender\SignatureNotValidFault
- Sender\UnknownDocumentTypeFault
- Receiver\MessagePersistencyFault
- Receiver\XsltTransformationFault
- Receiver\InternalSystemFailureFault
- Receiver\DispatchingNotPossibleFault
- Receiver\DispatchingDestinationUnknownFault
- Receiver\DispatchingFailedFault
- Receiver\DispatchingDeniedFault
- Receiver\ValidationFault

SchemaValidationFault is returned when the schema validation of the message fails. (It is only if the validation fails not if some implementation specific detail fails.)

SchematronValidationFault is returned when the schematron validation of the message fails.

SignatureNotValidFault is returned when the signature used is invalid.

UnknownDocumentTypeFault is returned if the type of the document is unknown for the receiver.

RaspMessagePersistencyFault is returned if a message is received but it was not successfully written to disk.

XsltTransformation is returned if the incoming message cannot be transformed.

InternalSystemFailureFault is returned if an internal error happens at the receiver. An example could be a failure when loading a specific schema from disk.

DispatchingNotPossibleFault

The endpoint could not contact the final dispatching destination, or the final destination will not accept the message. This fault indicates a non-temporary dispatching problem.

>

DispatchingDestinationUnknownFault

The final dispatching destination is not known for some reason, e.g. if a gateway is tasked with routing a message to an endpoint it does not know the physical address of.

DispatchingFailedFault

The endpoint tries to dispatch the message to its final destination, but the dispatch process fails for some reason. Please try again later.

DispatchingDeniedFault

The use of this endpoint to dispatch a message to its final destination has been denied. (+ a description of why)

ValidationFault

A generic validation fault which is not limited to schema or schematron validation.

Example Fault

The below (non-normative) example shows sample fault:

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:m="http://www.example.org/timeouts"
              xmlns:xml="http://www.w3.org/XML/1998/namespace">
  <env:Body>
    <env:Fault>
      <env:Code>
        <env:Value>env:Sender</env:Value>
        <env:Subcode>
          <env:Value>SchemaValidationFault</env:Value>
        </env:Subcode>
      </env:Code>
      <env:Reason>
        <env:Text xml:lang="en">
Failed to schema validate the message body.
The document with rootnode \"Invoice\" and namespace
\"urn:oasis:names:specification:ubl:schema:xsd:Invoice-2\" failed schema
validation.
The element 'Invoice' in namespace
'urn:oasis:names:specification:ubl:schema:xsd:Invoice-2' has invalid child
element 'IssueTime' in namespace
'urn:oasis:names:specification:ubl:schema:xsd:CommonBasicComponents-2'. List of
possible elements expected: 'IssueDate' in namespace
'urn:oasis:names:specification:ubl:schema:xsd:CommonBasicComponents-2'.
          </env:Text>
        </env:Reason>
      </env:Fault>
    </env:Body>
  </env:Envelope>
```

OIO RASP Ports



This section describes what ports need to be sent over, and listened to, to receive documents via OIO RASP.

To minimize the risk that SMEs will have problems sending caused by closed ports on a firewall, RASP connections over the HTTP protocol should be made on the standard HTTP port (80) or an alternate HTTP port (8008 or 8080).



RSP-1023: OIO RASP HTTP connections SHOULD be made on port 80, 8080 or 8008

References

>

The OIO RASP profile builds on the following other OIO profiles which are available from IT & Telestyrelsen's web site (<http://www.itst.dk>).

- > **OIO Basic Profile 1.1**
- > **OIO Basic Security Profile 1.1**
- > **OIO Reliable Messaging Profile 1.1**
- > **OIO SMTP/MIME Base64 Transport Binding for SOAP**

Appendix A - Requirements Checklist

>

The following table summarizes the requirements of the profile:

ID	Specification
RSP-1001	OIO RASP services MUST conform to the rules of the OIO BP 1.1 except for the overrides and extensions specified in OIO RASP 1.2.
RSP-1002	An Envelope MUST conform to the structure specified in SOAP 1.2 Section 5, "SOAP Message Construct". SOAP 1.1 envelopes MUST NOT be used.
RSP-1003	<p>The SOAP body of the response message of the application level web service request/response that does not contain a SOAP fault SHOULD be empty.</p> <p>The receiver of a SOAP body of the response message of the application level web service request/response that does not contain a SOAP fault SHOULD ignore any possible contents of the SOAP body.</p>
RSP-1004	A Message MAY be sent using SMTP in accordance with the OIO SOAP SMTP binding.
RSP-1005	OIO RASP conformant services MUST conform to the rules of the OIO Basic Security Profile 1.1.
RSP-1006	All messages exchanged via the OIO RASP profile MUST be encrypted according to the rules defined in OIO Basic Security Profile 1.1. The entire payload of the message (SOAP body) MUST be encrypted.
RSP-1007	<p>All messages exchanged via the OIO RASP profile MUST be signed according to the rules defined in the OIO Basic Security Profile. Signing MUST happen before encryption and MUST include OIO RASP custom SOAP headers, OIO Reliable Messaging SOAP headers, WS-Addressing headers and the SOAP body.</p> <p>User-defined (non-RASP) headers are not required to be signed.</p> <p>Unsigned messages or headers MAY be ignored by the receiver.</p>

RSP-1008	The signature MUST be made with a private key bound to an OCES Company certificate or OCES Function certificate. The latter is recommended.
RSP-1009	<p>The sender of a message MUST include the sender certificate using an embedded security token as described in OIO BSP.</p> <p>The receiver of a message MUST in replies reference the certificates of the sender and receiver using a security token reference as described OIO BSP.</p>
RSP-1010	A SecurityTokenReference element MUST have a KeyIdentifier with the value of "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509SubjectKeyIdentifier", and an EncodingType of "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary".
RSP-1011	<p>The following WS-Security algorithm identifiers must be used:</p> <ul style="list-style-type: none">• Symmetric keys must be encrypted using the algorithm identified by "http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p".• Symmetric keys must be digested using the algorithm identified by "http://www.w3.org/2000/09/xmldsig#sha1".• The signature must use the canonicalization algorithm identified by "http://www.w3.org/2001/10/xml-exc14n#".• The signature must be created using the method identified by "http://www.w3.org/2000/09/xmldsig#rsa-sha1".
RSP-1012	OIO RASP messages MUST be sent reliably according to the rules of OIO RM 1.1. The RM protocol SHOULD be used in "exactly once" mode with delivery of messages in order.
RSP-1013	Exponential backoff SHOULD be enabled when possible
RSP-1014	Reliable Messaging acknowledgement messages MUST be signed according to the rules defined in OIO Basic Security Profile 1.1.
RSP-1015	The namespace custom OIO RASP SOAP header blocks is "http://rep.oio.dk/oiosi.ehandel.gov.dk/xml/schemas/2007/09/01/"

RSP-1016	Custom OIO RASP SOAP header blocks SHOULD have a mustUnderstand property which MAY be set to “true”.
RSP-1017	<p>OIO RASP messages must include SenderPartyIdentifier, ReceiverPartyIdentifier and MessageIdentifier header blocks.</p> <p>OIO RASP messages SHOULD include SenderPartyIdentifierType and ReceiverPartyIdentifierType header blocks.</p>
RSP-1018	OIO RASP SOAP faults MUST define a Code and a Reason. They SHOULD NOT define a Node or a Role but CAN define a Detail element, if detailed error messages in XML formatting are needed.
RSP-1019	<p>The Code element of a OIO RASP fault MUST have a top level code that is either Sender - if the fault is due to the sender of the faulted message - or Receiver - if the fault is due to an error at the receiving side.</p> <p>Furthermore a OIO RASP fault MUST have at least one sub-code, describing what type of fault it is. If the fault was send because of an exception thrown the first level of sub-code SHOULD correlate to the name of the exception type</p>
RSP-1020	The Reason SHOULD contain text explaining why the fault occurred, which SHOULD include the error messages found in any type of exception object that might have caused the fault to be sent. Reason MUST NOT include stack traces from such exceptions or from elsewhere, due to security issues related to exposing the internal structure of web services.
RSP-1021	<p>OIO RASP faults returned MUST be encrypted and signed in the same way regular SOAP messages would, whenever possible.</p> <p>If, however, the fault originates from a place in the communications stack where signing and encryption is not possible (for example at transport level, where no credentials are available) the unsigned, unencrypted fault should still be sent.</p>

>

- RSP-1022** The following fault codes **MUST** be used:
- Sender\SchemaValidationFault
 - Sender\SchematronValidationFault
 - Sender\SignatureNotValidFault
 - Sender\UnknownDocumentTypeFault
 - Receiver\MessagePersistencyFault
 - Receiver\XsltTransformationFault
 - Receiver\InternalSystemFailureFault
 - Receiver\DispatchingNotPossibleFault
 - Receiver\DispatchingDestinationUnknownFault
 - Receiver\DispatchingFailedFault
 - Receiver\DispatchingDeniedFault
 - Receiver\ValidationFault

RSP-1023 OIO RASP HTTP connections **SHOULD** be made on port 80, 8080 or 8008

Appendix B - OIO RASP Reliable Messaging Properties

>

The OIO RASP profile (through the OIO Reliable Messaging Profile) builds on the WS-Reliable Messaging specification which defines the wire protocol for reliable message exchanges. It is important to realize that this protocol does not by itself provide any delivery assurances unless it is supplemented by other mechanisms at the end-points (for example persistence). Thus, when OIO RASP is used in contexts where such delivery assurance properties are needed, additional mechanisms must be implemented. Depending on the chosen mechanisms, different kinds of assurances can be achieved, but the wire protocol remains the same.

OIO RASP acknowledgements should be seen as “transmission” acknowledgements sent by the infrastructure. They do not guarantee that the message was delivered to the receiver’s business application or even processed correctly within some context. It is analogous to a situation in the physical world where a reception desk signs a receipt for a package on behalf of the intended receiver. For example, if the business message was an “order” then the acknowledgement cannot be regarded as an “order confirmation”. Note further that the acknowledgements do not relate to the content of the message - they simply identify the message id that was received. This means that technical non-repudiation of the message content is not achieved by OIO RASP acknowledgements.

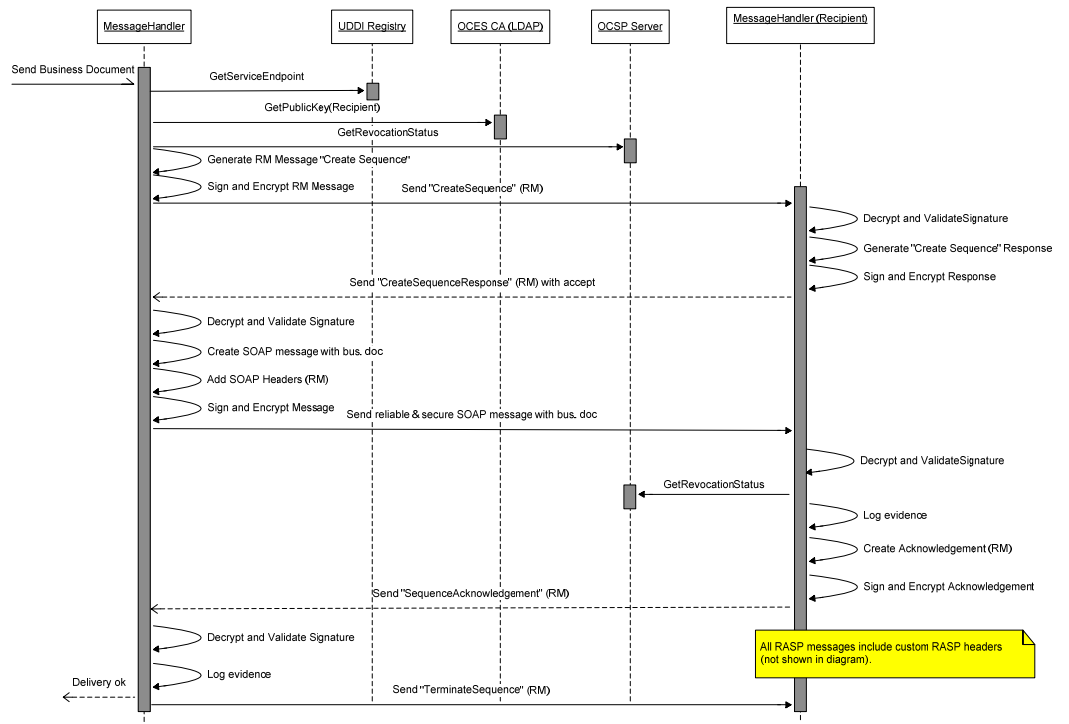
Another important consideration is message duplicates. With OIO RASP, message duplicates may occur at business application level even if duplicates are filtered by the underlying reliable messaging sessions (between CreateSequence and TerminateSequence). This is because a session may time out before acknowledgements are received or processed which can cause a message to be re-sent in a new session.

If stronger reliable messaging or non-repudiation properties are required, these must be implemented at the business application layer above OIO RASP. This could for example be receipts that a message was accepted or processed with a signature over the message content (e.g. a signed order confirmation). Another possibility is in the area of duplicates. Here OIO RASP defines a custom SOAP header containing a message ID which is defined to be unique per business message. Hence, this message ID would be suitable for implementing filtering of duplicates on top of OIO RASP if this is required by the business application.

Appendix C - Sequence Diagram



The (non-normative) sequence diagram below illustrates interactions between a sender and receiver in a scenario where one message is sent and acknowledged successfully. The figure serves illustrative purposes and some steps (related to last message and sequence termination) have been omitted for clarity.



Appendix D - OIO RASP usage in NemHandel

>

Introduction

This appendix defines the usage of the OIO Reliable Asynchronous Secure Profile version 1.2 (hereafter, "Profile" or OIO RASP) within the context of NemHandel.

About NemHandel

To put shortly, NemHandel is the concept of OIOUBL business documents sent using RASP. All public sector institutions in Denmark is capable of receiving invoices using NemHandel.

Sending OIO business documents

SOAP actions to use in RASP calls

The Action header used in SOAP should be set according to what type of business document is sent. The following section will describe actions (and expected reply actions) to use per document type.

The sections are constructed like

Root namespace:

`http://.../...`

Document type

Request action

Reply action

And actions are constructed as root namespace + request/response action.

OIOUBL

Root namespace

<http://rep.oio.dk/oiosi.ehandel.gov.dk/xml/schemas/2007/09/01/>

ApplicationResponse

ApplicationResponse201Interface/SubmitApplicationResponseRequest

ApplicationResponse201Interface/SubmitApplicationResponseResponse

CreditNote

CreditNote201Interface/SubmitCreditNoteRequest

CreditNote201Interface/SubmitCreditNoteResponse

Invoice

Invoice201Interface/SubmitInvoiceRequest

Invoice201Interface/SubmitInvoiceResponse

Order

Order201Interface/SubmitOrderRequest

>

Order201Interface/SubmitOrderResponse

OrderResponseSimple

OrderResponseSimple201Interface/SubmitOrderResponseSimpleRequest
OrderResponseSimple201Interface/SubmitOrderResponseSimpleResponse

Reminder

Reminder201Interface/SubmitReminderRequest
Reminder201Interface/SubmitReminderResponse

Catalogue

CatalogueResponse201Interface/SubmitCatalogueResponseRequest
CatalogueResponse201Interface/SubmitCatalogueResponseRequestResponse

CatalogueDeletion

CatalogueDeletionResponse201Interface/SubmitCatalogueDeletionResponseRequest
CatalogueDeletionResponse201Interface/SubmitCatalogueDeletionResponseResponse

CatalogueRequest

CatalogueRequestResponse201Interface/SubmitCatalogueRequestResponseRequest
CatalogueRequestResponse201Interface/SubmitCatalogueRequestResponseResponse

CataloguePricingUpdate

CataloguePricingUpdateResponse201Interface/SubmitCataloguePricingUpdateResponseRequest
CataloguePricingUpdateResponse201Interface/SubmitCataloguePricingUpdateResponseResponse

CatalogueItemSpecification Update

CatalogueItemSpecificationUpdateResponse201Interface/SubmitCatalogueItemSpecificationUpdateResponseRequest
CatalogueItemSpecificationUpdateResponse201Interface/SubmitCatalogueItemSpecificationUpdateResponseResponse

OrderCancellation

OrderCancellationResponse201Interface/SubmitOrderCancellationResponseRequest

OrderResponse

OrderResponseResponse201Interface/SubmitOrderResponseResponseRequest

OrderChange

OrderChangeResponse201Interface/SubmitOrderChangeResponseRequest

Statement

StatementResponse201Interface/SubmitStatementResponseRequest

OIOXML

Root namespace

<http://rep.oio.dk/oiosi.ehandel.gov.dk/xml/schemas/2007/09/01/>

Invoice (0.7)

Invoice07Interface/SubmitInvoice07Request
Invoice07Interface/SubmitInvoice07Response

CreditNote (0.7)

Creditnote07Interface/SubmitCreditNote07Request
Creditnote07Interface/SubmitCreditNote07Response

Invoice (0.7.1) "læs ind"

Invoice07pipInterface/SubmitInvoice07pipRequest
Invoice07pipInterface/SubmitInvoice07pipResponse

CreditNote (0.7.1) "læs ind"

Creditnote07pcpInterface/SubmitCreditNote07pcpRequest
Creditnote07pcpInterface/SubmitCreditNote07pcpResponse

