The Bank of Russia Standard

STO BR NPS-6.1-2020

FINANCIAL MESSAGES IN THE NPS.
DATA EXCHANGE RULES

Introduction date: 2019-09-30

Official Publication

Moscow
2020

Preamble

ACCEPTED AND ENACTED by The Bank of Russia's order of 20 September 2019, No OD-2191, "On the enactment of the Bank of Russia Standard STO BR NPS-1.4-2019 “Financial messages in the NPS. Data Exchange Rules”.

Contents

Introduction

This Standard contains the recommendations on how to implement a data exchange framework in accordance with ISO 20022[1] to the transfer of funds the National Payment System (hereinafter – the NPS).

[1] ISO 20022 international Standard “Financial Services” — Universal financial industry message scheme

The Bank of Russia Standard

FINANCIAL MESSAGES IN THE NPS.
DATA EXCHANGE RULES

Introduction date: 2019-09-30

1. Scope

This Standard is recommended to developers of information and program software, information systems for organizing data exchange between banks and their customers.

Inter-bank exchange of financial messages in the course of service provision can rely either on in-house data exchange rules or the provisions included in this Standard.

Provisions of this Standard are applied on a voluntary basis, unless regulatory acts of the Bank of Russia or terms of contracts make some provisions obligatory.

2. Terms and definitions

Terms of Bank of Russia Standards STO BR NPS-1.1-2020 “Financial Messages in the NPS. General Terms”, Federal Law No. 63-FZ, dated 6 April 2011, “On Electronic Signature” and terms below are used in this Standard.

OSI model	–	The basic reference model of open systems interconnection in line with GOST R ISO/IEC 7498-1-99 ‘State Standard of the Russian Federation. Information Technology. Open Systems Interconnection. Basic Reference Model. Part 1. Basic Model’;
Message	–	A set of structured information (in digital form) being the object of exchange between Data Exchange Participants through means of communication;
Business Message	–	A digital ISO 20022-compliant message used by Data Exchange Participants to communicate application data;
Delivery Message	–	An envelope compliant with the SOAP 1.2 specification, its headers containing structures related to message signing and encryption and its body including the transported Business Message;
Message Delivery Mode	–	Group of settings for the values determining Message Delivery Characteristics;
Message Delivery System	–	A mechanism enabling the receipt, Delivery and delivery of Delivery Messages between Data Exchange Participants;
Data Exchange Participant	–	An actor who creates, processes, receives or sends digital messages;
Digest	–	A limited-length string generated through special transformations of source data in line with GOST R 34.11-2012;
Deflate	–	Lossless data compression algorithm;
Base64	–	Binary-to-text encoding and decoding Standard;
Validation	–	Validation of incoming data in line with pre-set rules.

3. General provision

In accordance with ISO 20022-6 “Message Delivery Characteristics,” data exchange within the NPS is carried out in the following layers:

- The Business Layer is the top layer setting the rules and scenarios for the Business Message exchange, as well as the structures of Business Messages used in exchange processes within particular NPS relationship models. The Business Layer is equivalent to adding Level 9 to the OSI model.
- The Delivery Layer is the layer setting the digital message exchange rules, independent of the Business Layer. The Delivery Layer is equivalent to adding Layer 8 to the OSI model.
- The Application Layer is the lowest layer equivalent to Level 7 in the OSI model.

This Standard contains the recommendations for the NPS-based data exchange in each of the above layers. Compliance with these recommendations will help ensure functional interoperability between ISO 20022-compliant data systems interacting within the NPS.

4. Business Layer

Data exchange in the Business Layer involves Business Message transport. A Business Message is a combination of a business header and the message content described in the ISO 20022 Standards of the NPS.

Business Messages must contain business data only and may not include any information on the Message Delivery System, the mechanisms for message addressing and sending, Delivery and receipt, as well as any other information specific to the Delivery Layer or the Application Layer.

Business Messages must contain all the information needed for their correct interpretation by Data Exchange Participants and be comprehensible beyond the context of the Message Delivery Envelope (Section 5 of the Standard).

Business Messages must be prepared as XML documents and must comply with the XML schemas and the rules for filling in documents that are stipulated in the ISO 20022 Standards of the NPS.

4.1. Data exchange rules

The rules for data exchange in the Business Layer are stipulated by the ISO 20022 Standards of the NPS describing the models of relations between money transfer participants and messages used within these models.

The description of data exchange in the Business Layer must specify the Message Delivery Modes that determine the groups of the Message Delivery Characteristics described in Part 1 “Metamodel” of ISO 20022-1.[2] The Message Delivery Characteristics are given for reference purposes in Table 4.1 included in this Standard.

The Message Delivery Modes set in the Business Layer must be implemented in the Delivery Layer and/or the Application Layer.

Table 4.1. Message Delivery Characteristics

Characteristics	Description
DeliveryAssurance	Takes one of the following values: • AT_LEAST_ONCE • EXACTLY_ONCE • AT_MOST_ONCE
SenderAsynchronicity	Takes one of the following values: • ENDPOINT_SYNCHRONOUS (the endpoint is synchronous) • CONVERSATION_SYNCHRONOUS (data exchange is synchronous) • ASYNCHRONOUS
ReceiverAsynchronicity	Takes one of the following values: • ENDPOINT_SYNCHRONOUS (the endpoint is synchronous)

[2] www.iso20022.org

	• CONVERSATION_SYNCHRONOUS (data exchange is synchronous) • ASYNCHRONOUS
MessageDeliveryOrder	Takes one of the following values: • EXPECTED_CAUSAL_ORDER (the order remains for all Senders and Receivers) • FIFO_ORDERED (the order remains only for each pair of the Sender and the Receiver) • UNORDERED (the order does not remain)
MessageDeliveryWindow	Maximum time during which the Delivery Message may be delivered without regard to the order of priority. The values must be larger or equal to zero.
MessageSendingWindow	Maximum time during which the Delivery Message may be sent without regard to the order of priority. The values must be larger or equal to zero.
MessageCasting	Takes one of the following values: • UNICAST • MULTICAST • BROADCAST • ANYCAST
BoundedCommunicationDelay	Maximum time during which the message must be delivered. The value must set a time period in line with ISO 8601
MessageValidationOnOff	Takes one of the following values: • VALIDATION_ON (messages are validated by the Message Delivery System) • VALIDATION_OFF (messages are not validated by the Message Delivery System)
MessageValidationResults	Takes one of the following values: • REJECT • REJECT_AND_DELIVER • DELIVER
MessageValidationLevel	The message validation layer required by the Message Delivery System. Takes one of the following values: • NO_VALIDATION (there was no message validation) • SYNTAX_VALID (the message syntax was validated) • SCHEMA_VALID (the message matches the XML schema) • MESSAGE_VALID (the message complies with … rules) • RULE_VALID (the message complies with business rules) • MARKET_PRACTICE_VALID (the message complies with market practice) • BUSINESS_PROCESS_VALID (the message complies with the business process) • COMPLETELY_VALID (the message is completely valid)
Durability	Takes one of the following values: • DURABLE • PERSISTENT • TRANSIENT
MaximumClockVariation	Maximum variance from Coordinated Universal Time (UTC) for the supported Message Delivery Mode
MaximumMessageSize	Maximum size of the Delivery Message in kilobytes (any positive integer larger than zero)

4.2. Business Message structure

A Business Message must have the following structure:

```
<BusinessMessage xmlns="urn:cbrf:iso:20022:business-message">
  <AppHdr xmlns="urn:iso:std:iso:20022:tech:xsd:head.001.001.01">
    <!-- Header content -->
  </AppHdr>
  <Document xmlns="urn:iso:std:iso:20022:tech:xsd:xxxx.nnn.nnn.nn">
    <!-- Document content -->
  </Document>
</BusinessMessage>
```

The Business Application Header (BAH) structure is used as the header of the Business Message. The structure is specified in the document 'ISO 20022 Business Application Header. Message Definition Report' (version dated 1 June 2010).[3] The framework rules are stipulated in the document 'ISO 20022 Business Application Header Message Usage Guide Version 1.8' (version issued in April 2016). The structure of the Business Message header and the rules on how to fill it out for the purposes of the NPS are described in Table 4.2.

Table 4.2. Structure of the Business Message header

Application attribute	Creation rules
Sender	The Sender's identifier is specified in the AppHdr/Fr/* attribute. It is also necessary to specify the identification schema in the */SchmeNm/Cd attribute (when identification codes registered in ISO 20022 are used) or in the */SchmeNm/Prtry attribute (when in-house identification codes are used). Below are the key identifier types: LEI (Legal Entity Identifier) a unique 20-digit alphanumeric identifier (code) of a legal entity that is assigned in accordance with the standard ISO 17442 OGRN (primary state registration number), identification schema – OGRN TIN, identification schema – TXID Russian BIC, identification schema – RUCBC UIMA, identification schema – RUCB Data Exchange Participants may use another identifier accepted by them that ensures unambiguous identification of the participant in the exchange chain. Space, Line Feed, Carriage Return, or Tabulation may not be used to create the Sender's identifier and the identifier type.
Receiver	The Receiver's identifier is specified in the AppHdr/To/* attribute. The requirements for the Receiver's identifiers are similar to those for the Sender's identifiers.
Message Identifier	The message identifier is specified in the AppHdr/BizMsgIdr attribute. The message identifier must be unique for an individual Data Exchange Participant. The Universally Unique Identifier (UUID) algorithms in line with the RFC 4122 specification (see Table 5.1) must be used to create message identifiers. When the UUID is specified in the AppHdr/BizMsgIdr attribute, it is not allowed to use the urn:uuid: namespace identifier and the symbols "-" in the string representation of the UUID, in order to ensure compliance with the requirements for the BAH identifier length. Identifier example: "f81d4fae7dec11d0a76500a0c91e6bf6"
Message Type	The Business Message type is specified in the AppHdr/MsgDefIdr attribute. It is only permitted to use the message types supported by the NPS. Example: pain.001.001.07
Business Service	The Business Service is specified in the AppHdr/BizSvc attribute. This is an optional attribute, and therefore the rules on how to fill it out are to be set by Data Exchange Participants.

[3] www.iso20022.org

Message Creation Date and Time	The date and time (GMT) of the message creation are to be specified in the AppHdr/CreDt attribute.

5. Delivery Layer

5.1. Data exchange rules

Data exchange in the Delivery Layer is carried out through Delivery Messages containing Business Messages. The structure of Delivery Messages and the rules for their creation are stipulated in Section 5.2 included in this Standard.

Data exchange scenario:

1. The Sender creates a Business Message.
2. The Sender packs the created Business Message inside the Delivery Message.
3. The Sender sends the Delivery Message to the Receiver.
4. The Receiver receives the Delivery Message.
5. The Receiver unpacks the Delivery Message and extracts the Business Message from it.
6. If the unpacking procedure fails, the Receiver creates a failure message according to Section 5.2.5 included in this Standard and sends it to the Sender.
7. The Receiver initiates validation of the Business Message if the validation requirement is set in the Business Layer.
8. The Receiver processes the Business Message after successful validation (or when validation is not performed).
9. If validation fails, the Receiver creates a failure message according to Section 5.2.5 included in this Standard and sends it to the Sender.

The Delivery Message packing (unpacking) procedures comprise encryption (decryption) operations, as well as signing with a digital signature. The rules on how to perform these procedures are described in Sections 5.2.3 and 5.2.4 included in this Standard.

The Sender may send Delivery Messages to more than one Receiver depending on the Message Casting parameter set in the Business Layer.

The process of message Delivery from the Sender to the Receiver may use one or more Message Delivery Systems supporting message Delivery within their delivery area and capable, inter alia, to cast Delivery Messages.

To identify the Receiver (Receivers), Message Delivery Systems may use both Delivery Message attributes and Business Message attributes (e.g. values of Business Message header elements). The content of the Business Message may not be altered when it is processed by the Message Delivery System.

Data exchange in the Delivery Layer is performed in accordance with the WS-I Basic Profile Version 2.0 specification.

5.2. Message Delivery Envelope structure and creation rules

5.2.1. General rules for Message Delivery Envelope creation

The structure of the Delivery Message and the rules for its creation must be based on the specifications given in Table 5.1.

Table 5.1. Specifications

Abbreviated name	Full name
SOAP 1.2	SOAP Version 1.2 Part 1: Messaging Framework (Second Edition). W3C Recommendation 27 April 2007. http://www.w3.org/TR/soap12-part1
WS Security 1.1.1	Web Services Security: SOAP Message Security Version 1.1.1. OASIS Standard. 18 May 2012. http://docs.oasis-open.org/wss-m/wss/v1.1.1/os/wss-SOAPMessageSecurity-v1.1.1-os.html
XML-binary Optimized Packaging	XML-binary Optimized Packaging. W3C Recommendation 25 January 2005. http://www.w3.org/TR/2005/REC-xop10-20050125
XML Encryption 1.1	XML Encryption Syntax and Processing Version 1.1. W3C Recommendation 11 April 2013 https://www.w3.org/TR/xmlenc-core/
XML 1.0	Extensible Markup Language (XML) 1.0 (Fifth Edition). W3C Recommendation 26 November 2008.

Abbreviated name	Full name
	http://www.w3.org/TR/2008/REC-xml-20081126
RFC 2045	RFC 2045: Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies. http://tools.ietf.org/rfc/rfc2045.txt
RFC 3986	RFC 3986. Uniform Resource Identifier (URI): Generic Syntax. http://tools.ietf.org/html/rfc3986
RFC 4122	RFC 4122. A Universally Unique IDentifier (UUID) URN Namespace. http://www.ietf.org/rfc/rfc4122.txt
RFC 4648	RFC 4648: The Base16, Base32, and Base64 Data Encodings. http://tools.ietf.org/rfc/rfc4648.txt

Namespaces listed in Table 5.2 must be used to describe the structure of digital messages.

Table 5.2. Namespaces

Prefix	Namespace
ds	http://www.w3.org/2000/09/xmldsig#
soap	http://www.w3.org/2003/05/soap-envelope
xenc	http://www.w3.org/2001/04/xmlenc#
wsa	http://schemas.xmlsoap.org/ws/2004/08/addressing
wsse	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd
wsu	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd
head	urn:iso:std:iso:20022:tech:xsd:head.001.001.01
msg	urn:cbrf:iso:20022:business-message
vr	urn:cbrf:iso:20022:validation-results

The Message Delivery Envelope has the following structure:

```
<soap:Envelope
    xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
    xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd">
  <soap:Header>
    <!—Message Envelope Header -->
  </soap:Header>
  <soap:Body wsu:Id="BusinessMessage">
    <!-- Business Message -->
  </soap:Body>
</soap:Envelope>
```

The UTF-8 encoding must be used to create digital messages.

5.2.2. Binary attachments

A Business Message may contain binary attachments within its structure. When Business Messages are sent within Delivery Messages, binary attachments may be transmitted in the message body in the Base64 encoding (according to RFC 4648) or created as separate MIME parts.

The packing of binary attachments into separate MIME parts is intended to streamline Delivery processes and reduce the size of messages with binary attachments.

When binary attachments are transmitted as separate MIME parts, a reference to the MIME part is generated in the message body according to the XML-binary Optimized Packaging specification.

MIME parts of messages are intended to streamline message Delivery processes. Messages must be processed in line with the XOP processing model of the XML-binary Optimized Packaging specification.

Binary attachments are recommended to be packed as MIME parts when a binary attachment exceeds 100 Kb.

5.2.3. Message signing and digital signature validation

The signing with a digital signature and its validation must comply with the WS Security 1.1.1 specification. The recommendations on how to create and validate a digital signature are provided in Annex 1.

The structure of the wsse:Security element containing a digital signature is described in Tables 5.3 and 5.4.

Table 5.3. Structure of the wsse:Security element containing a digital signature

No.	Element	Description	Value domain	M
a)	Sign interpretations requirement @soap:mustUnderstand	The sign that the Message Receiver must process the element according to the specification. The attribute has a fixed value in line with the description in Annex 1	Boolean value	1
1	Digital signature verification key certificate wsse:BinarySecurityToken	The certificate of the key for validation of the digital signature used to sign the message	Base64-encoded string of characters	1
a)	Identifier @wsu:Id	A unique identifier of the element in the message	Document element identifier	1
b)	Type @ValueType	The Standard for the certificate of the key for digital signature validation. The attribute has a fixed value in line with the description in Annex 1	Uniform Resource Identifier	1
c)	Encoding @EncodingType	The encoding scheme for the certificate of the key for digital signature validation. The attribute has a fixed value in line with the description in Annex 1	Uniform Resource Identifier	1
2	Digital signature ds:Signature	The digital signature of the message in line with Table 5.4	The data structure is described in Table 5.4	1

Table 5.4. Digital signature structure (ds:Signature)

No.	Element	Description	Value domain	M
1	Signed information ds:SignedInfo	A set of elements being signed that contain the Business Message hash and a set of instructions on how to process the hash and the signature	Determined by the value domains of nested elements	1
1.1	Canonicalization method ds:CanonicalizationMethod	Instructions on canonicalisation	None	1
a)	Algorithm @Algorithm	Canonicalisation algorithm. The attribute has a fixed value in line with the description in Annex 1	Uniform Resource Identifier	1
1.2	Signature method ds:SignatureMethod	The element containing the signature creation and validation algorithm	None	1
a)	Algorithm @Algorithm	The identifier of the signature creation and validation algorithm. The attribute has a fixed value in line with the description in Annex 1	Uniform Resource Identifier	1
1.3	Reference ds:Reference	A reference to the data for which the hash is generated	Determined by the value domains of nested elements	1
a)	Identifier @URI	The hashed area identifier. The attribute has a fixed value in line with the description in Annex 1	Document element identifier	1
1.3.1	Transforms ds:Transforms	Instructions on how the hashed area must be transformed	Determined by the value domains of nested elements	1
1.3.1.1	Transform ds:Transform	An element containing a single instruction on how the hashed area must be transformed	Determined by the value domains of nested elements	1
a)	Algorithm @Algorithm	The transformation algorithm identifier. The attribute has a fixed value in line with the description in	Uniform Resource Identifier	1

No.	Element	Description	Value domain	M
		Annex 1		
1.3.1.2	Digest method ds:DigestMethod	Data on the hash method	None	1
a)	Algorithm @Algorithm	The digest algorithm identifier. The attribute has a fixed value in line with the description in Annex 1	Uniform Resource Identifier	1
1.3.1.3	Digest value ds:DigestValue	The hash value	Base64-encoded string of characters	1
2	Signature value ds:SignatureValue	The signature value	Determined by the value domains of nested elements	1
3	Key Info ds:KeyInfo	Data on the Public Key used to validate the signature	Determined by the value domains of nested elements	1
3.1	Security token reference wsse:SecurityTokenReference	Data on the key location	Determined by the value domains of nested elements	1
3.1.1	Reference wsse:Reference	The element containing the key location identifier	None	1
a)	Identifier @URI	The key location identifier. The attribute has a fixed value in line with the description in Annex 1	Document element identifier	1
b)	Type @ValueType	The key certificate Standard. The attribute has a fixed value in line with the description in Annex 1	Uniform Resource Identifier	1

5.2.4. Message encryption and decryption

Messages must be encrypted and decrypted in accordance with the WS Encryption 1.1 specification. Messages are to be encrypted after they are signed.

Message encryption procedure.

1. The wsse:Security element is added to the header of the Message Delivery Envelope. This element must precede the wsse:Security element containing a digital signature (if any).
2. The certificate of the Public Key used to encrypt the message is added to the wsse:Security/ wsse:BinarySecurityToken element. The certificate complies with the X.509 Standard (specified in the ValueType attribute) and is encoded according to the Base64 Standard (specified in the EncodingType attribute).
3. The wsse:Security/xenc:EncryptedKey/ds:KeyInfo element specifies the reference to the Public Key certificate.
4. The Business Message in the soap:Body element is encrypted according to the XML Encryption 1.1 specification. The encrypted message is put in the xenc:EncryptedData element replacing the original Business Message.
5. The wsse:Security/xenc:EncryptedKey/xenc:ReferenceList element includes the reference to the encrypted message.

Message decryption procedure.

1. If the header of the Message Delivery Envelope contains the wsse:Security/xenc:EncryptedKey element, the message is encrypted and requires decryption.
2. The Encrypted Key is determined for the Public Key certificate referenced in the wsse:Security/ xenc:EncryptedKey/ds:KeyInfo element in order to decrypt the message.
3. The data referenced in the wsse:Security/xenc:EncryptedKey/xenc:ReferenceList element are decrypted with the Encrypted Key. The result is put into the message to replace the encrypted data.
4. The wsse:Security element containing information on message encryption is removed from the message.
5. If the message decryption fails, a failure message is generated according to Section 5.2.5 included in this Standard.

The structure of the wsse:Security element containing information on message encryption is described in Tables 5.5 and 5.6.

Table 5.5. Structure of the wsse:Security element with message encryption data

No.	Element	Description	Value domain	M
a)	Sign interpretations requirement @soap:mustUnderstand	The sign that the Message Receiver must process the element in line with the specification. The attribute has a fixed "true" value	Boolean value	1
1	Public key certificate wsse:BinarySecurityToken	The certificate of the Public Key used to encrypt the message	Base64-encoded string of characters	1
a)	Identifier @wsu:Id	A unique identifier of the element in the message	Document element identifier	1
b)	Type @ValueType	The Standard for the Public Key certificate. The attribute has a fixed value, similar to the one described in Annex 1	Uniform Resource Identifier	1
c)	Encoding @EncodingType	The encoding scheme for the Public Key certificate. The attribute has a fixed value, similar to the one described in Annex 1	Uniform Resource Identifier	1
2	Information about encrypted data xenc:EncryptedKey	Information on encrypted data	The data structure is described in Table 5.6	1

Table 5.6. Structure of information on encrypted data (xenc:EncryptedKey)

No.	Element	Description	Value domain	M
1	Encryption method xenc:EncryptionMethod	Information on the encryption method	Determined by the value domains of nested elements	1
a)	Algorithm @Algorithm	The encryption algorithm identifier	Uniform Resource Identifier	1
2	Key Info ds:KeyInfo	Information on the Public Key used for encryption	Determined by the value domains of nested elements	1
2.1	Security token reference wsse:SecurityTokenReference	Data on the key location	Determined by the value domains of nested elements	1
2.1.1	Reference wsse:Reference	The element containing the key location identifier	None	1
a)	Identifier @URI	The key location identifier. The attribute has a fixed value in line with the description in Annex 1	Document element identifier	1
b)	Type @ValueType	The key certificate Standard. The attribute has a fixed value in line with the description in Annex 1	Uniform Resource Identifier	1
3	Reference list xenc:ReferenceList	The list of references to encrypted data	Determined by the value domains of nested elements	1
3.1	Data reference xenc:DataReference	The reference to encrypted data	None	1
a)	Identifier @URI	The encrypted data identifier	Document element identifier	1

The structure of encrypted data is described in Table 5.7.

Table 5.7. Encrypted data structure (xenc:EncryptedData)

No.	Element	Description	Value domain	M
a)	Identifier @Id	The encrypted data identifier	Document element identifier	1
1	Encryption method xenc:EncryptionMethod	Data on the encryption method	None	1
a)	Algorithm @Algorithm	The encryption algorithm identifier	Uniform Resource Identifier	1
2	Cipher data xenc:CipherData	The element containing encrypted data	Determined by the value domains of nested elements	1

2.1	Cipher value xenc:CipherValue	Encrypted data	Base64-encoded string of characters	1

5.2.5. Generation of failure messages

Failure messages are generated in line with the SOAP 1.2 specification.

Failure messages are transmitted in the body of the Message Delivery Envelope and must be its only element.

Table 5.8. Failure data structure (soap:failure)

No.	Element	Description	Value domain	M
1	Failure code soap:Code	Formalised data on the failure code	Determined by the value domains of nested elements	1
1.1	Value soap:Value	The value of the failure class	Listing of soap: failureCodeEnum in line with the SOAP 1.2 specification	1
1.2	Failure code soap:Subcode	Data on the failure code in line with its class	Determined by the value domains of nested elements	1
1.2.1	Value soap:Value	The failure code value. Standard failure codes are listed in Table 5.9	String of characters	1
2	Reason soap:Reason	Information on the failure	Determined by the value domains of nested elements	1..*
2.1	Text soap:Text	Description of the failure	String of characters	1
a)	lang @xml:lang	The identifier of the language used to describe the failure		1
3	Failure detail soap:Detail	Extra details on the failure	Determined by the value domains of nested elements	0..1

The soap:Code/soap:Value element is filled in according to the SOAP 1.2 specification.

Standard failure codes are listed in Table 5.9.

Table 5.9. Standard failures

Failure class	Failure code	Failure text
soap:Sender	wsse:UnsupportedSecurityToken	It is recommended to use Standard failure codes specified in the WS-Security Standard
soap:Sender	wsse:UnsupportedAlgorithm	
soap:Sender	wsse:InvalidSecurity	
soap:Sender	wsse:InvalidSecurityToken	
soap:Sender	wsse:FailedAuthentication	
soap:Sender	wsse:FailedCheck	
soap:Sender	wsse:SecurityTokenUnavailable	
soap:Sender	wsse:MessageExpired	
soap:Sender	vr:InvalidSyntax	Message syntax is invalid
soap:Sender	vr:InvalidStructure	Message structure does not match XML schema
soap:Sender	vr:MessageRulesViolated	Message violates structural rules
soap:Sender	vr:BusinessRulesViolated	Message violates business rules
soap:Sender	vr:MarketPracticeViolated	Message violates market practice
soap:Sender	vr:BusinessProcessViolated	Message violates business process

If a message violates the rules (vr:MessageRulesViolated, vr:BusinessRulesViolated), the soap:Failure/soap:Detail element must include the vr:Result elements containing information on validation results.

A separate vr:Result element is generated for each validated element and for each rule. Information on validation failures must be included in the message. For debugging purposes, a failure message may include information on other validation results (valid, irrelevant, absent, unsupported).

Table 5.10.Structure of information on validation results

No.	Element	Description	Value domain	M
1	Validation results vr:Result	Validation results	Determined by the value domains of nested elements	1..*
1.1	Validation result kind vr:Kind	The codename for the validation result kind	Takes one of the following values: • valid • error • irrelevant • absent • unsupported	1
1.2	Description of the validation result vr:Description	Description of the validation result	String of characters	1
1.3	Rule code vr:RuleCode	The rule code	String of characters	1
1.4	Rule description vr:RuleDescription	Description of the rule	String of characters	1
1.5	Rule specification vr:RuleOcl	The rule specification in the formal Object Constraint Language (OCL)	String of characters	0..1
1.6	Element vr:Element	The message element validated	Determined by the value domains of nested elements	1..*
1.6.1	Path vr:Path	The path to the message element in the form of an XPath expression	String of characters	1
1.6.2	Description vr:Description	A description of the path to the element in the Russian language	String of characters	0..1

6. Application Layer

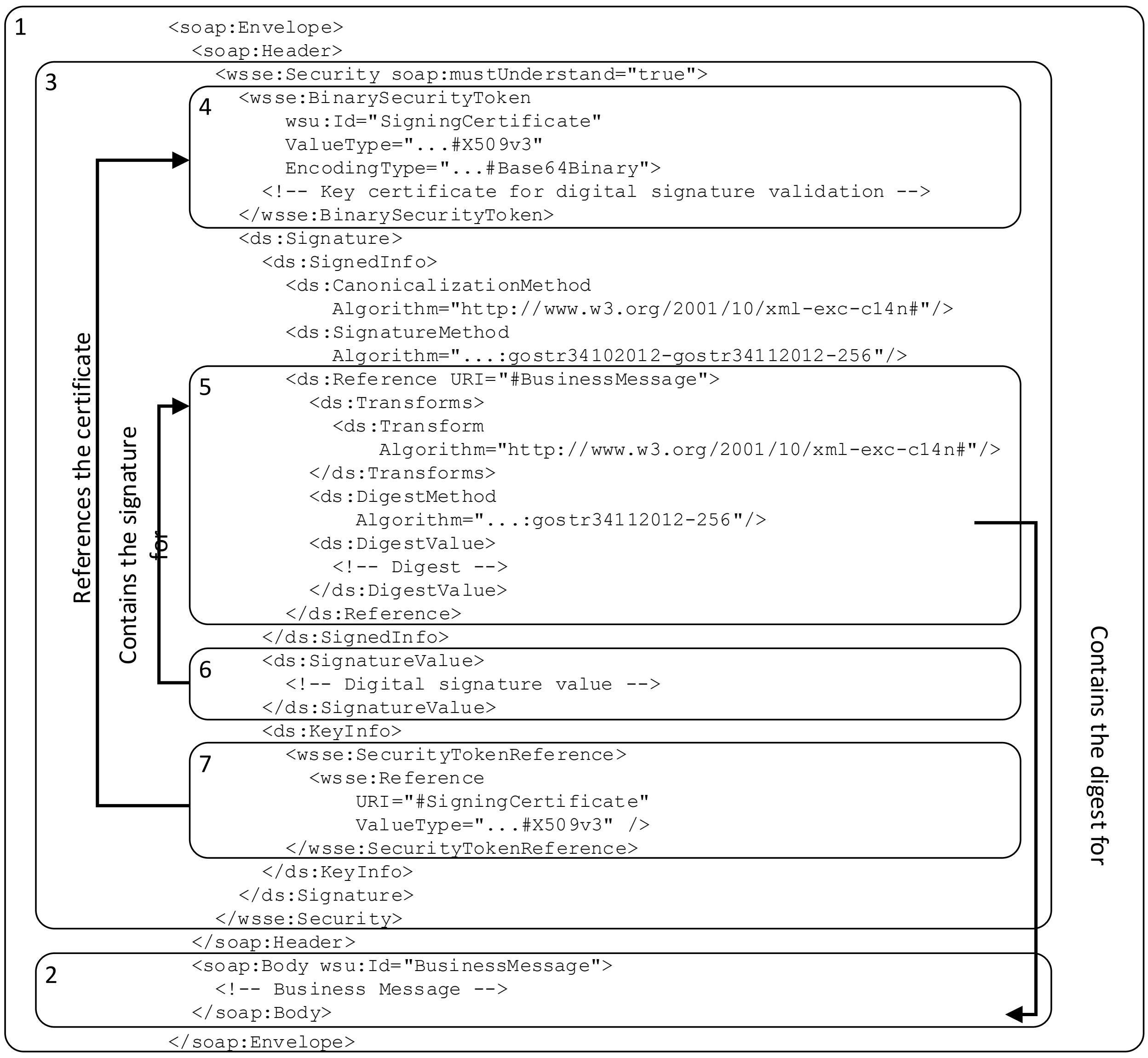
As a data Delivery protocol for the Application Layer, Data Exchange Participants choose one of the protocols supporting the required Message Delivery Characteristics set in the Business Layer.

The following protocols are recommended:

– HTTPS: Hypertext Transfer Protocol - HTTP/1.1 (RFC 2616) jointly with the HTTP Over TLS (RFC 2818) specification;
– AMQP: ISO/IEC 19464:2014 Information technology - Advanced Message Queuing Protocol (AMQP) v1.0 specification

Annex 1

The signing of data transported in the Message Transport Envelope with a digital signature is carried out as follows (see Chart A.1). For brevity, the Chart omits namespaces, and values of some attributes are abbreviated.



```
<soap:Envelope>
  <soap:Header>
    <wsse:Security soap:mustUnderstand="true">
      <wsse:BinarySecurityToken
          wsu:Id="SigningCertificate"
          ValueType="...#X509v3"
          EncodingType="...#Base64Binary">
        <!-- Key certificate for digital signature validation -->
      </wsse:BinarySecurityToken>
      <ds:Signature>
        <ds:SignedInfo>
          <ds:CanonicalizationMethod
              Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
          <ds:SignatureMethod
              Algorithm="...:gostr34102012-gostr34112012-256"/>
          <ds:Reference URI="#BusinessMessage">
            <ds:Transforms>
              <ds:Transform
                  Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
            </ds:Transforms>
            <ds:DigestMethod
                Algorithm="...:gostr34112012-256"/>
            <ds:DigestValue>
              <!-- Digest -->
            </ds:DigestValue>
          </ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>
          <!-- Digital signature value -->
        </ds:SignatureValue>
        <ds:KeyInfo>
          <wsse:SecurityTokenReference>
            <wsse:Reference
                URI="#SigningCertificate"
                ValueType="...#X509v3" />
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>
    </wsse:Security>
  </soap:Header>
  <soap:Body wsu:Id="BusinessMessage">
    <!-- Business Message -->
  </soap:Body>
</soap:Envelope>
```

Chart A.1. Message signing scheme

Step 1 is to generate the Message Delivery Envelope (Step 1 is not directly related to the signature creation, but given in this algorithm as a preparation for building XML structures that will carry the signature and the signed data).

Step 2 is to put a Business Message into the soap:Body element of the Message Delivery Envelope. An identifier that must be unique within the envelope and its content is specified in the @wsu:ID attribute of the soap:Body element. It is recommended to use the value “BusinessMessage” as the identifier for the message body. The digital signature created will reference the signed data by this identifier.

Step 3 is to create a block of the wsse:Security data in the header of the Message Delivery Envelope. The @soap:mustUnderstand attribute with a “true” value is set for the element. It obliges the Message Receiver to process this data block. If the digital signature processing fails, the Message Receiver must notify the Sender thereof with a relevant failure message.

Step 4 is to put the wsse:BinarySecurityToken element containing the certificate of the key for digital signature validation into the wsse:Security data block. The certificate identifier which is unique within the envelope and its content is specified in the @wsu:Id attribute. It is recommended to use the value “SigningCertificate” for identifying the certificate. The digital signature created will reference the certificate by this identifier. The certificate must comply with the X.509 Standard and be encoded using the Base64 encoding Standard.

The certificate Standard is specified in the @ValueType attribute with the value “http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3”. The certificate encoding scheme is specified in the @EncodingType attribute with the value “http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary”.

Step 5 is to generate the Business Message hash. For this purpose, the ds:Signature/ds:SignedInfo/ds:Reference element is created in the wsse:Security data block. The @URI attribute of this element must contain the reference to the Business Message signed (if the recommended identifier was set at Step 2, the @URI attribute must contain the reference “#BusinessMessage”). If binary attachments are transmitted as MIME, they must be then transferred to the Business Message in line with the algorithm described in the XML-binary Optimized Packaging specification. The Business Message is then canonicalised using the Exclusive XML Canonicalization Version 1.0 algorithm. A 256-bit digest is calculated for the received XML document fragment in accordance with GOST R 34.11-2012 and is specified in the ds:DigestValue element.

The canonicalisation algorithm is specified in the ds:Transforms/ds:Transform/@Algorithm attribute with the value “http://www.w3.org/2001/10/xml-exc-c14n#”. The hash calculation algorithm is specified in the ds:DigestMethod/@Algorithm attribute with the value “urn:ietf:params:xml:ns:cpxmlsec:algorithms:gostr34112012-256”.

Step 6 is to create a digital signature for the Business Message hash. Data on its canonicalisation algorithm (Exclusive XML Canonicalization Version 1.0) and the digital signature calculation algorithm (GOST 34.10-2012, 256 bit) are added into the ds:Signature/ds:SignedInfo element. These algorithms are then used to canonicalise the ds:Signature/ds:SignedInfo element, and a digital signature is calculated for the received XML document fragment. The value is specified in the ds:SignatureValue element.

The canonicalisation algorithm is specified in the ds:CanonicalizationMethod/@Algorithm attribute with the value “http://www.w3.org/2001/10/xml-exc-c14n#”. The digital signature calculation algorithm is specified in the ds:SignatureMethod/@Algorithm attribute with the value “urn:ietf:params:xml:ns:cpxmlsec:algorithms:gostr34102012-gostr34112012-256”.

Step 7 is to create a reference to the certificate of the key for digital signature validation in the ds:KeyInfo element in the information on the digital signature. The wsse:SecurityTokenReference element is used for this purpose. The reference to the certificate is specified in the wsse:Reference/@URI attribute (if the recommended identifier was set at Step 4, the @URI attribute must contain the reference “#SigningCertificate”). The certificate Standard is specified in the wsse:Reference/@ValueType attribute with the value “http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3”.

The digital signature of the message is validated as follows (see Chart A.2). For brevity, the scheme omits namespaces, and values of some attributes are abbreviated.

Step 1 is to check the Message Delivery Envelope header for the wsse:Security/ds:Signature element. If the element is present, the message is deemed to have been signed, and the applied digital signature must be validated.

Step 2 is to validate the certificate of the digital signature Public Key in the wsse:BinarySecurityToken element. The certificate must be correct, and the ds:KeyInfo/wsse:SecurityTokenReference element must contain a reference thereto.

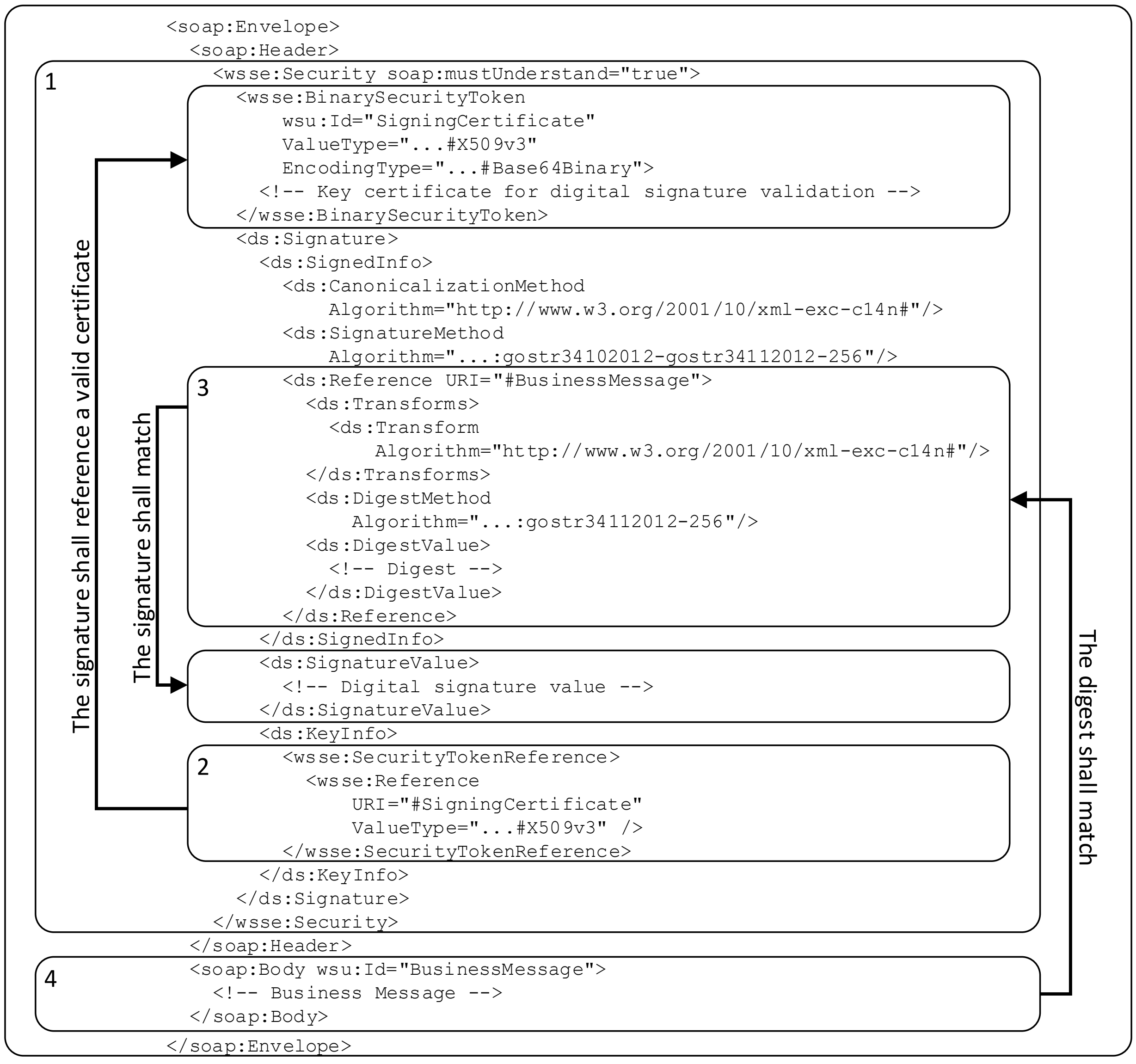
Step 3 is to validate the digital signature. For this purpose, the ds:SignedInfo element is canonicalised using the Exclusive XML Canonicalization Version 1.0 algorithm. A 256-bit value of the digital signature is calculated in line with GOST 34.10-2012 for the received XML document fragment. If the result matches the value of the ds:SignatureValue element, the signature is valid.

The value of the ds:SignedInfo/ds:CanonicalizationMethod/@Algorithm attribute is validated: it must match the canonicalisation method used and have the value “http://www.w3.org/2001/10/xml-exc-c14n#”. The value of the ds:SignedInfo/ds:SignatureMethod/@Algorithm attribute is validated: it must match the algorithm used to calculate the digital signature and have the value ‘urn:ietf:params:xml:ns:cpxmlsec:algorithms:gostr34102012-gostr34112012-256’.

Step 4 is to check the Business Message for integrity. If binary attachments are transmitted as MIME, they must be transferred to the Business Message before the check for integrity according to the algorithm described in the XML-binary Optimized Packaging specification. The Business Message is then canonicalised using the Exclusive XML Canonicalization Version 1.0 algorithm. A 256-bit digest is calculated for the received XML document fragment in line with GOST R 34.11-2012. If the value matches that of the ds:SignedInfo/ds:Reference/ds:DigestValue element, the Business Message is deemed to be integral.

The value of the ds:SignedInfo/ds:Reference/ds:Transforms/ds:Transform/@Algorithm attribute is validated: it must match the canonicalisation method used and have the value “http://www.w3.org/2001/10/xml-exc-c14n#”. The value of the ds:SignedInfo/ds:Reference/ds:DigestMethod/@Algorithm attribute is validated: it must match the digest calculation algorithm and have the value ‘urn:ietf:params:xml:ns:cpxmlsec:algorithms:gostr34112012-256’.

If the digital signature validation fails, a failure message is generated according to Section 5.2.5 included in this Standard.



```
<soap:Envelope>
  <soap:Header>
    <wsse:Security soap:mustUnderstand="true">
      <wsse:BinarySecurityToken
          wsu:Id="SigningCertificate"
          ValueType="...#X509v3"
          EncodingType="...#Base64Binary">
        <!-- Key certificate for digital signature validation -->
      </wsse:BinarySecurityToken>
      <ds:Signature>
        <ds:SignedInfo>
          <ds:CanonicalizationMethod
              Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
          <ds:SignatureMethod
              Algorithm="...:gostr34102012-gostr34112012-256"/>
          <ds:Reference URI="#BusinessMessage">
            <ds:Transforms>
              <ds:Transform
                  Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
            </ds:Transforms>
            <ds:DigestMethod
                Algorithm="...:gostr34112012-256"/>
            <ds:DigestValue>
              <!-- Digest -->
            </ds:DigestValue>
          </ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>
          <!-- Digital signature value -->
        </ds:SignatureValue>
        <ds:KeyInfo>
          <wsse:SecurityTokenReference>
            <wsse:Reference
                URI="#SigningCertificate"
                ValueType="...#X509v3" />
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>
    </wsse:Security>
  </soap:Header>
  <soap:Body wsu:Id="BusinessMessage">
    <!-- Business Message -->
  </soap:Body>
</soap:Envelope>
```

1
3
2
4
The signature shall reference a valid certificate
The signature shall match
The digest shall match

Chart A.2. Digital signature validation scheme

Annex 2

Business Message XML schema.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns="urn:cbrf:iso:20022:msg"
    targetNamespace="urn:cbrf:iso:20022:msg"
    xmlns:bah="urn:iso:std:iso:20022:tech:xsd:head.001.001.01"
    elementFormDefailure="qualified">
  <xs:import namespace="urn:iso:std:iso:20022:tech:xsd:head.001.001.01"/>
  <xs:element name="BusinessMessage" type="BusinessMessageType" />
  <xs:complexType name="BusinessMessageType">
    <xs:sequence>
      <xs:element ref="bah:AppHdr"/>
      <xs:any processContents="strict" />
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

Annex 3

Example of a message with a digital signature.

```
<soap:Envelope
    xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
  <soap:Header>
    <wsse:Security
        xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd"
        xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd"
        xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
        soap:mustUnderstand="true">
      <wsse:BinarySecurityToken
          wsu:Id="SigningCertificate"
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3"
          EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
soap-message-security-1.0#Base64Binary">
        <!-- Key certificate for digital signature validation -->
      </wsse:BinarySecurityToken>
      <ds:Signature>
        <ds:SignedInfo>
          <ds:CanonicalizationMethod
              Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
          <ds:SignatureMethod
              Algorithm=
"urn:ietf:params:xml:ns:cpxmlsec:algorithms:gostr34102012-gostr34112012-256"/>
          <ds:Reference URI="#BusinessMessage">
            <ds:Transforms>
              <ds:Transform
                  Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
            </ds:Transforms>
            <ds:DigestMethod
                Algorithm=
"urn:ietf:params:xml:ns:cpxmlsec:algorithms:gostr34112012-256"/>
            <ds:DigestValue>
              <!-- Digest -->
            </ds:DigestValue>
          </ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>
          <!-- Signature Value -->
        </ds:SignatureValue>
        <ds:KeyInfo>
          <wsse:SecurityTokenReference>
            <wsse:Reference
                URI="#SigningCertificate"
                ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-x509-token-profile-1.0#X509v3" />
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>
    </wsse:Security>
  </soap:Header>
  <soap:Body wsu:Id="BusinessMessage">
    <!-- Business Message -->
  </soap:Body>
</soap:Envelope>
```

Annex 4

Example of an encrypted message.

```
<soap:Envelope
    xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
  <soap:Header>
    <wsse:Security
        xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd"
        xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd"
        xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
        xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
        soap:mustUnderstand="true">
      <wsse:BinarySecurityToken
          wsu:Id="EncryptionCertificate"
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v3"
          EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
soap-message-security-1.0#Base64Binary">
         <!-- Public Key certificate -->
      </wsse:BinarySecurityToken>
      <xenc:EncryptedKey>
        <xenc:EncryptionMethod
            Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
        <ds:KeyInfo>
          <wsse:SecurityTokenReference>
            <wsse:Reference
                URI="#EncryptionCertificate"
                ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-x509-token-profile-1.0#X509v3" />
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
        <xenc:ReferenceList>
          <xenc:DataReference URI="#EncryptedMessage"/>
        </xenc:ReferenceList>
      </xenc:EncryptedKey>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <xenc:EncryptedData Id="EncryptedMessage">
      <xenc:EncryptionMethod
          Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
      <xenc:CipherData>
        <xenc:CipherValue>
          <!-- Encrypted Message -->
        </xenc:CipherValue>
      </xenc:CipherData>
    </xenc:EncryptedData>
  </soap:Body>
</soap:Envelope>
```