SSEK Version 2.0

Secure Web Services for Business Critical Communication

10 May 2006

Mats Andersson, Skandia Liv
Peter Danielsson, Skandia Liv
Gustaf Nyman, Skandia Liv

Abstract

SSEK 2.0 specifies the design of secure web services for information exchange between organisations. SSEK has been proven compliant in practical usage with existing business, security and legal requirements for electronic business communication. Use of SSEK improves the organisation's capacity to create secure and compatible business communication with other organisations.

SSEK is based on a number of accepted standards for Internet-based web services.

Compared to SSEK 1.1, SSEK 2.0 contains a number of improvements in functionality and updates of the standards on which SSEK is based for the current version.

Contents

1 Introduction 1
1.1 SSEK Working Group 2
1.2 Definitions 2
1.3 Notational conventions 2
1.4 Namespaces and prefixes 2
2 SSEK Basics 3
3 Message Structures 3
3.1 SSEK 3
3.2 Standardised fault messages 5
3.3 Standardised receipt 7
3.4 Handling of attachments 7
4 Message Flows 8
4.1 One-way message flow 8
4.2 Synchronous message flow 8
4.3 Asynchronous message flow with delivery (push) 8
4.4 Asynchronous message flow with retrieval (pull) 8
4.5 Fault situations related to message flows 9
5 Security 10
5.1 Transport security 10
5.2 Message security 10
5.3 X509 certificate 10
5.4 Message signature 11
6 Metadata 11
6.1 SSEK Policy Language 12
7 SSEK 2.0 as a Component of Business Agreements 13
8 Differences Compared to SSEK 1.1 13
9 References 15
Appendix 1: XML Schema for SSEK 16
Appendix 2: XML Schema for SSEK Policy Language 18

1 Introduction

This document defines how secure web services are produced and consumed over the Internet according to SSEK 2.0 The specification is the result of needs identified by the insurance sector but may be useful in any context where the following factors are important:

- Technically secure information exchange between organisations
- Interoperability between implementations and platforms
- Capacity to securely implement agreed web services in practical web service utilisation

The SSEK specification is based on guidelines associated with accepted specifications from W3C, IETF, OASIS, WS-I and others combined with several specific additions unique to SSEK. SSEK defines the following:

- Addressing at the organisational level
- Handling of message flows and resending
- Standard structures for faults and receipts
- Guidelines for security, messages, metadata, attachments and interoperability.

The objective of this document is to describe as accurately as possible what SSEK is and what it is not. The document is intended primarily for developers of system software for SSEK.

The authors disclaim all liability for how SSEK is used in practice. The authors guarantee nothing of evidentiary value in any dispute regarding circumstances or conditions documented according to SSEK.

1.1 SSEK Working Group

SSEK 2.0 was prepared by the SSEK Working Group, whose members are:

- Mats Andersson, Skandia Liv
- Lars Boshuis, SEB
- Peter Danielsson, Skandia Liv
- Johan Lidö, SEB
- Gustaf Nyman, Skandia Liv

1.2 Definitions

The following terms are used in the document as defined below:

Terms	Meaning
Message	SOAP message [SOAP11][BP11] according to SSEK
Sender	Party that sends a message according to SSEK
Receiver	Party that receives a message according to SSEK
Web service	Web service that supports SSEK
Producer	Party that provides a value-creating web service according to SSEK
Consumer	Party that uses a web service according to SSEK
SOAP header	The soap:Header element according to [SOAP11] and [BP11]
Header element	A child element under the SOAP header
SOAP body	The soap:Body element according to [SOAP11] and [BP11]
Payload	The payload is the part of the message under the SOAP body that constitutes the essential content of the message
SOAP fault	Message whose payload is a soap:Fault according to [SOAP11]
Fault code	The faultcode element in a SOAP fault

1.3 Notational conventions

The keys words SHOULD, MAY, MUST and MUST NOT in this document are to be interpreted as follows:

Keyword	Meaning
SHOULD	This word mean that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.
MAY	This word mean that an item is truly optional.
MUST	This word mean that the definition is an absolute requirement of the specification.
MUST NOT	This phrase mean that the definition is an absolute prohibition of the specification.

1.4 Namespaces and prefixes

The document uses prefixes as defined in the table.

Prefix	Namespace
soap	http://schemas.xmlsoap.org/soap/envelope/
wsse	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd
wsse11	http://docs.oasis-open.org/wss/oasis-wsswssecurity-secext-1.1.xsd
wsu	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd
ssek	http://schemas.ssek.org/ssek/2006-05-10/
ssekp	http://schemas.ssek.org/ ssek/2006-05-10/policy

2 SSEK Basics

SOAP 1.1 is used for transporting messages according to SSEK.

B001 Messages MUST be compatible with SOAP 1.1 Envelopes [SOAP11]

The WS-I Basic Profile Version 1.1 [BP11] Specification is the result of practical experience with problems of interoperability among web service implementations. Problems with interoperability arise primarily from lack of clarity in fundamental specifications such as [SOAP11] and [WSDL]. [BP11] provides concrete guidelines on how web services should be designed to ensure effective interoperability with other systems.

B002 WS-I Basic Profile Version 1.1 [BP11] including errata [BP11ERR] MUST be followed except in cases explicitly described in this specification.

Several aspects of SSEK differ from [BP11]:

B003 UDDI MUST NOT be used to describe web services.

B004 Only document-literal binding MUST be used.

B005 Only one element MUST appear under the SOAP body.

WS-Addressing [WSA][WSASB] defines the use of endpoints to identify senders and receivers. This information is not necessary for addressing with SSEK, where addressing takes place instead at a more abstract level based on sender and receiver identificiations. Nonetheless, addressing according to SSEK and WS-Addressing are fully interoperable.

B006 WS-Addressing [WSA][WSASB] MAY be used with SSEK.

3 Message Structures

3.1 SSEK

The SSEK element is a header element used to carry information about the sender and receiver of a message and the message flow in which the message is included.

TX001 The SSEK element is a header element that MUST be included in messages with the exception of SOAP faults.

TX002 The SSEK element MUST have the attribute soap:mustUnderstand with value 1.

Simplified description of the SSEK element. For an exact description see the appended schema.

```
<ssek:SSEK ssek:AsynchMethod="AsynchPush|AsynchPull" soap:mustUnderstand="1" {any}?>
  <ssek:SenderId ssek:Type="CN|DN|ORGNR|APP">...</ssek:SenderId>
  <ssek:ReceiverId ssek:Type="CN|DN|ORGNR|APP">...</ssek:ReceiverId>
  <ssek:TxId>...</ssek:TxId>?
</ssek:SSEK>
```

/ssek:SSEK

This element is a header element and MUST be placed below the SOAP header.

/ssek:SSEK/@ssek:AsynchMethod

AsynchMethod is an optional attribute that indicates which type of asynchronous communication the consumer wants to use. The attribute may be used when the relevant web service is asynchronous and must contain one of the values defined in the web service policy. May have the value AsynchPush or AsynchPull.

/ssek:SSEK/ssek:SenderId

SenderId states the sender's identity. How the contents of SenderId should be interpreted is governed by the ssek:Type attribute.

/ssek:SSEK/ssek:SenderId/@ssek:Type='CN'

Type is an optional attribute that defines the contents of the SenderId element. Type may have any of the values CN, DN, ORGNR or APP. The default value CN is presumed if the Type attribute is missing. The web service policy may govern which value must be used.

/ssek:SSEK/ssek:ReceiverId

ReceiverId states the receiver's identity. How the contents of ReceiverId should be interpreted is governed by the ssek:Type attribute.

/ssek:SSEK/ssek:ReceiverId /@ssek:Type='CN'

Type is an optional attribute that defines the contents of the ReceiverId element. Type may have any of the values CN, DN, ORGNR or APP. The default value CN is presumed if the Type attribute is missing. The web service policy may govern the value that must be used.

/ssek:SSEK/ssek:TxId

TxId is an optional element that identifies the message flow that contains the message by means of a unique identifier, UUID. The web service policy governs whether or not TxId must be used.

/ssek:SSEK/@{any}

The SSEK element permits other attributes.

3.1.1 SenderId and ReceiverId

Information exchange with SSEK takes place primarily between organisations, which also constitute the main addressable units in SSEK. Organisations are identified and addressed according to the following table.

Type	Meaning	Description
CN	Common Name	Reference to a certificate that identifies the organisation
DN	Distinguished Name	Reference to a certificate that identifies the organisation
ORGNR	Organisation number	An organisation's registration number
APP	Application	Usually used internally by organisations to identify specific applications or systems; should not be used between organisations

Senders and receivers can be identified in several ways, but the recommendation for maximum security is to identify the sender and receiver as Distinguished Names (DN), where DN refers to the X509 certificate used for digitally signing a message. This relates the stated sender identity (SenderId) to the certificate used for digital message signature.

TX003 If digital signature is used, SenderId and ReceiverId MUST be stated as a Common Name (CN) or Distinguished Name (DN).

TX004 If digital signature is used, SenderId and ReceiverId MUST have the same Common Name (CN) or Distinguished Name (DN) as the certificates used to digitally sign the message and reply, respectively.

TX005 If CN is used without a digital signature, CN MUST correspond to the CN that would have been specified in an intended certificate for the organisation.

3.1.2 TxId

TxId identifies a message flow (see Chapter 4) between two or more parties. A message flow is a series of related messages. Syntactically, a TxId is a unique identifier [UUID]. Usage of TxId for a web service is governed by the web service policy.

TX006 TxId MUST be used to identify messages when required.

TxId must always be created according to accepted algorithms for generating UUID, which guarantees that new TxIds will not coincide with existing TxIds.

TX007 TxId MUST be generated according to a standardised algorithm [UUID].

TxId may be reused under certain circumstances.

TX008 If several messages belong together in one message flow, one TxId MUST be used to identify all messages.

If a message does not belong to the message flow it identifies, a fault message must be returned from the receiver.

TX009 If a message does not belong to the current message flow indicated by its TxId or does not start a new message flow, the receiver MUST return a SOAP fault with the fault code ssek:IncorrectContext.

If a message is not processed by the receiver, the sender may under certain conditions resend the message with the same TxId. Resending is allowed if a SOAP fault has been received by the sender and the fault code is not ssek:Timeout, or if the policy allows resending on timeout.

TX010 Messages MAY be resent when SOAP fault is received and fault code is not ssek:Timeout.

TX011 Messages MAY be resent if there is no response or fault code is ssek:Timeout if the web service policy states ssekp:ResendAllowedOnTimeout.

TX012 When SOAP fault is returned, the producer MUST roll back all transactions.

SSEK 2.0 thus permits resends under certain conditions, but there is a consequence for producers. When a SOAP fault is returned (other than with fault code ssek:Timeout), if any transactions have been executed, all current transactions MUST be rolled back. Resend after timeout may also be allowed with ssekp:ResendAllowedOnTimeout for web services that are not updating.

SSEK does not require the receiver to check that TxId is not reused after a message flow has been ended.

3.2 Standardised fault messages

All faults in a synchronised message flow, technical and application-specific faults, are to be returned as SOAP faults according to [BP11] and [SOAP].

F001 All faults identified by the receiver MUST be returned to the sender as SOAP faults.

F002 If SOAP fault is returned to the sender, permanent changes MUST NOT be implemented at the receiver and no transactions may be executed. If the receiver cannot guarantee this, ssek:Timeout MUST be returned to the sender.

Domain-specific fault codes may be defined for application faults. They must be defined in XML Schema and the data type must be QName.

F003 The data type for application-specific fault codes MUST be QName.

Additional fault information may need to be communicated to the receiver in some cases. The FaultData structure exists for the purpose.

F004 FaultData MAY can be incorporated in SOAP fault under the detail element to describe fault situations in detail.

Simplified description of FaultData. For an exact description see the appended schema.

```
<ssek:FaultData>
  <ssek:FaultingMessage>...</ssek:FaultingMessage>?
  <ssek:TxId>...</ssek:TxId>?
  <ssek:FaultItems>?
    <ssek:FaultItem>
      <ssek:Code>...</ssek:Code>
      <ssek:Description>...</ssek:Description>?
      <ssek:Location>...</ssek:Location>?
      <ssek:System>...</ssek:System>?
    </ssek:FaultItem>*
  </ssek:FaultItems>
  {any}*
</ssek:FaultData>
```

/ssek:FaultData
The element may be placed in SOAP fault under the detail element and used to describe a fault more exhaustively than possible in the fault code and fault string element in the SOAP fault.

/ssek:FaultData/ssek:FaultingMessage
If a query message contains a fault, this element can be used to return the faulting message to the sender. The message must be incorporated as text data.

/ssek:FaultData/ssek:TxId
The optional TxId element may be used to return TxId (if applicable) from a query message.

/ssek:FaultData/ssek:FaultingItems/ssek:FaultingItem
The FaultingItem element may occur more than once to describe faults in an incoming message.

/ssek:FaultData/ssek:FaultingItems/ssek:FaultingItem/ssek:Code
The element must include a fault code for the specific fault.

/ssek:FaultData/ssek:FaultingItems/ssek:FaultingItem/ssek:Description
The element is used to describe the specific fault.

/ssek:FaultData/ssek:FaultingItems/ssek:FaultingItem/ssek:Location
Optional element that may be used to identify a faulting element in FaultingMessage with xpath or by other means.

/ssek:FaultData/ssek:FaultingItems/ssek:FaultingItem/ssek:System
Optional element to specify a faulting system.

/ssek:FaultData/{any}
If there is additional information not covered by other elements, a custom-defined element may be incorporated.

3.2.1 Fault Codes

The table below defines fault code values specific to SSEK.

Faultcode	**Description**
ssek:AsynchMethodUnsupported	The selected asynchronous method is not supported by the web service.
ssek:ContentInvalid	The message content is invalid.
ssek:IncorrectContext	A message that does not belong to the current message flow or does not start a new message flow has been received. See TX009.
ssek:MessageNotProcessed	The message was not processed because of a fault at the receiver.
ssek:NoResultAvailable	No result available for retrieval.
ssek:ReceiverIdUnknown	Receiver's identity unknown.
ssek:SenderIdUnknown	Sender's identity unknown.
ssek:Timeout	Receiver status has become unstable. Resending is not allowed unless allowed by the policy. Contact the receiving organisation.
ssek:TxIdInvalid	TxId is not syntactically correct.
ssek:TxIdMissing	TxId is missing.
ssek:TxIdNotAllowed	TxId is not allowed for this web service.
ssek:TxIdUnknown	TxId is unknown.
ssek:WebServiceUnavailable	The web service is not available.
ssek:WebServiceUnsupported	The web service is not supported by the receiver.

3.2.2 Fault Management in Asynchronous Message Flows with Delivery

SOAP fault cannot be used in asynchronous message flows to describe faults during the processing phase when the producer calls the consumer's web service. In these cases, the application-specific message sent from the producer to the consumer should contain information on any faults.

3.3 Standardised receipt

SSEK defines a standard message to confirm delivery of a message. It fulfils its function primarily in asynchronous message flows.

K001 A standardised receipt MUST be used to confirm delivery of messages.

Simplified description of standard receipt. For an exact description see the appended schema.

```
<ssek:Receipt {any}?>
  <ssek:ResponseCode>OK</ssek:ResponseCode>
  <ssek:ResponseMessage>...</ssek:ResponseMessage>?
  <ssek:RequestSignatureValue>...</ssek: RequestSignatureValue>?
  {any}*
</ssek:Receipt>
```

/ssek:Receipt/ssek:ResponseCode

The standardised receipt must be used to acknowledge delivery of a query message. If a fault occurs, SOAP fault must be used instead. For that reason, the ResponseCode element may contain only OK.

/ssek:Receipt/ssek:ResponseMessage

The optional ResponseMessage element may be used for more detailed receipt information.

/ssek:Receipt/ssek:RequestSignatureValue

If the query message is signed, the RequestSignatureValue must contain the signature for the query message, except when [WSS11] is used fully, in which case the signature from the query message is returned in the Security header. The response message, i.e. the Receipt, must be signed, which creates a technical chain of evidence from the signed query message to the signed receipt.

K002 If the query message is signed, the standard receipt MUST include the signature providing the policy element ssekp:RequestSignatureHandling has the value Receipt or ReceiptAndSignatureConfirmation for the web service or no policy is stated.

/ssek:Receipt/{any}

If there is a need that cannot be met by other elements in the standardised receipt, custom-defined information may be added.

3.4 Handling of attachments

Attachments consist of non-XML-based information incorporated in XML documents after BASE64 coding. One disadvantage to BASE64 coding is that the procedure uses extra computer and network resources. The solution to the problem is the XML-binary Optimized Packaging (XOP) convention. XOP serialises the XML document, including BASE64 coded attachments, to a MIME document where the attachments are stored in their original format in MIME parts.

In theory, this means a serialisation of the document's XML information set. In practice, it means attachments can be processed separately without BASE64 coding and incorporated directly into the XML document. We thus get an XML document made up of several MIME parts. For details see [XOP], [MTOM] and [SOAP11MTOM]. Note that in practice BASE64 coding cannot be avoided for signed messages.

A001 Attachments, both BASE64 coded and optimised according to [XOP], [MTOM] and [SOAP11MTOM] MUST be handled.

4 Message Flows

Message flow refers to the transfer of one or more messages between consumer and producer in order to exchange information.

SSEK defines four basic types of message flows:

- One-way message flow
- Synchronous message flow
- Asynchronous message flow with delivery (push)
- Asynchronous message flow with retrieval (pull)

The four types may be combined to form more complex message flows.

4.1 One-way message flow

A one-way message flow consists of a message sent from consumer to producer. The producer does not return any message, but instead normally ends the flow by returning HTTP return code 200 [SOAP].

M001 Producers MAY support one-way message flows. Consumers MUST support one-way message flows.

4.2 Synchronous message flow

A synchronous message flow consists of a message sent from consumer to producer with a response message sent synchronously by the producer. Synchronously means that the response is returned in the HTTP response.

M002 The producer and consumer MUST support synchronous message flows.

4.3 Asynchronous message flow with delivery (push)

An asynchronous message flow with delivery (AML) is constructed of two synchronous message flows and two web services. It is initiated with a synchronous message flow from the consumer to the producer. The producer delivers a message receipt to the consumer and begins processing the message. The roles are reversed when processing is completed: the producer initiates a synchronous message flow to the consumer and delivers the processing result. In this case, both the producer and the consumer must provide web services.

M003 It MUST be stated in the policy if a web service supports AML.

M004 If a consumer intends to initiate an AML, it MUST be stated using the attribute AsynchMethod in ssek:SSEK set to AsynchPush.

4.4 Asynchronous message flow with retrieval (pull)

An asynchronous message flow with retrieval (AMH) also consists of two synchronous message flows. The message flow is initiated with a synchronous message flow from the consumer to the producer. The producer delivers a message receipt to the consumer and begins processing the message.

After a period of time, the consumer initiates a synchronous message flow requesting the processing result. If the processing result is available, it is returned to the customer. Otherwise, a fault message is returned with fault code ssek:NoResultAvailable.

The consumer requests the result with the message PullMessage.

```
<ssek:PullMessage {any}?>
  {any}*
</ssek:PullMessage>
```

/ssek:PullMessage

The PullMessage element must be incorporated as payload and denotes a query about the processing result in an AMH.

/ssek:PullMessage/{any}

Custom-defined element may be incorporated for application-specific purposes.

/ssek:PullMessage/@{any}
Custom-defined attribute may be incorporated for application-specific purposes.

A PullMessage itself does not contain any information about the processing requested, as this is identified implicitly by the message flow's TxId.

The advantage of the AMH procedure compared to AML is that the consumer is simplified considerably because it does not have to provide a web service and can act solely as message sender.

M005 It MUST be stated in the policy if a web service supports AMH.

M006 If a consumer initiates an AMH, it MUST be stated with the attribute AsynchMethod in ssek:SSEK set to AsynchPull.

M007 The consumer MUST request the processing result with the message ssek:PullMessage.

M008 The fault code ssek:NoResultAvailable MUST be returned when no result is available in an AMH.

It must be noted that with AMH, the producer does not get a receipt to indicate that the consumer has received a processing response.

4.5 Fault situations related to message flows

Fault situations related to message flows:

M009 If the consumer has selected a type of message flow not supported by the web service, the producer MUST return the SOAP fault with the fault code ssek:AsynchMethodUnsupported.

5 Security

The following business needs have been identified regarding the security of communication over the Internet:

- Information must be protected from unauthorised reading or manipulation.
- It must be possible to reliably trace the information to the sender.

SSEK complies with the security aspects described in the following table in order to meet business needs for security. SSEK uses PKI (Public Key Infrastructure) [PKI] for the purpose.

Aspect	**Explanation**	**Support in SSEK**
Confidentiality	Messages cannot be read during transport by anyone other than the receiver.	SSL
Receiver authentication	The receiver's identity is authenticated for the sender.	SSL with server certificate
Sender authentication	The sender's identity is authenticated for the receiver.	SSL with (client) organisation certificate.
Integrity, correctness	The messages have not been changed during transport from sender to receiver.	Signing with stamp certificate[1].
Non-repudiation	The sender cannot deny that the contents of a message were created by the sender.	Signing with stamp certificate.

SSEK divides security aspects into transport security (TransportLevelSecurity) and message security (MessageLevelSecurity).

[1] Certificate identifying an organisation with the purpose of signing information

5.1 Transport security

SSEK specifies two levels of transport security:

- Confidentiality and receiver authentication (SSL)
- Security and sender/receiver authentication (SSLWithClientCertificate)

SSEK requires the use of transport security level SSL at minimum. Transport security level SSL means the information is protected against unauthorised reading and the receiver is identified. The transport security level SSL WithClientCertificate means that the sender is also identified. This may be used for instance when publishing a web service for which protection against unauthorised access is required.

S001 SSL with server authentication (SSL) MUST be used.

S002 SSL with client authentication (SSLWithClientCertificate) MAY be used.

S003 The transport security level MUST be stated in the web service policy.

5.2 Message security

SSEK defines two levels of message security:

- No message security (None)
- Non-repudiation and integrity (Signature)

Use of message security is optional. The message security level Signature is used if there is need to check and confirm that certain information was created by a particular sender.

S004 Message Signature MAY be used.

S005 The message security level MUST be stated in the web service policy.

S006 If signatures are used, all messages in the synchronous or asynchronous message flow MUST be signed.

5.3 X509 certificate

An X509 certificate [X599V3] is a document containing a relation between a public key and a number of attributes that typically describe an organisation. When a certificate is created and signed by a trusted organisation (Certificate Authority) that verifies that the certificate accurately describes the organisation, the certificate can be used to identify the organisation. When an organisation uses its private key, which is linked to the public key in the certificate (to sign a message for instance) the signature can later be used together with the certificate to prove that the organisation signed the message.

Certificates are used in SSEK to identify organisations and web servers. The certificate must be issued by a CA (Certificate Authority) that verifies its accuracy. A certificate may be revoked if for some reason it no longer uniquely identifies an organisation or web server.

The following guidelines apply to certificates used in SSEK.

S006 Certificates used for communication between two parties MUST be issued by a CA (Certificate Authority) and be of a type approved by both parties.

S007 The party that uses a certificate SHOULD perform a revocation check by accessing the CRL (Certificate Revocation List) published by the CA that issued the certificate or perform a revocation check using OCSP (Online Certificate Status Protocol) [OCSP] according to CA guidelines. If a revocation check is performed using CRL or OCSP, the check MUST be performed every time the certificate is used.

S008 Manual revocation is performed if a revocation check according to S007 is not performed. In a manual revocation, the party that wants to revoke a certificate contacts a responsible person at the other party as soon as possible, who then deletes the certificate so that it cannot be used. In this case, the certificate MUST also be revoked with the issuing CA.

5.4 Message signature

Message signature is based on underlying specifications from IETF, W3C, OASIS and WS-I. The IETF and W3C lay the foundations for certificate management, canonicalisation and signature of XML. OASIS defines the application to SOAP messages and WS-I increases interoperability between communicating systems.

OASIS has developed [WSS10] and an update to it [WSS11]. [WSS11] introduces a number of new functions but is backwards compatible with [WSS10]. The signature confirmation in [WSS11] is a particularly important function for SSEK that was implemented with a standard receipt in SSEK 1.1. For practical reasons, SSEK 2.0 requires support for [WSS10] but for future use allows support for signature confirmation in [WSS11] to be governed by policy.

SIG01 Signature of SSEK messages MUST be complying with [WSS10], [X509CTP], [C14], [XMLDSIG] and [BSP10] except as defined in this specification.

SIG02 wsu:Timestamp MUST be attached according to [WSS10] and [BSP10]. Both wsu:Created and wsu:Expired MUST be stated.

SIG03 ssek:SSEK, SOAP body, wsu:Timestamp and, where applicable for reply messages, wsse11:SignatureConfirmation MUST be signed.

SIG04 The X509 certificate used to sign the message MUST be attached in X509v3 format BASE64-coded in a wsse:BinarySecurityToken element.

SIG05 The X509 certificate that signed the message MUST be referred to directly via a wsse:SecurityTokenReference element.

SSEK makes certain restrictions related to use of the specifications.

SIG06 The parts of a message that are signed MUST be identified with wsu:Id attribute and referred to with a shorthand XPointer.

SIG07 The parts of a message that are transformed/signed MUST be transformed according to "http://www.w3.org/2001/10/xml-exc-c14n#" [C14N].

SIG08 The signature method "http://www.w3.org/2000/09/xmldsig#rsa-sha1" MUST be used. [XMLDSIG] .

SIG09 Message encryption MUST NOT be used.

6 Metadata

Metadata is information about messages. Two types of metadata are used:

- Information about what is to be communicated [XMLSCHEMA], [WSDL].
- Information about how it must be communicated [WS-POLICY].

Description of messages in SSEK is based on XML [XML] and XML Schema [XMLSCHEMA1] [XMLSCHEMA2].

MD01 XML Schema MUST be used to describe the payload.

WSDL 1.1 describes the messages a web service handles.

MD02 Web services MUST be described in WSDL 1.1 [WSDL].

MD03 Policies MUST be incorporated in WSDL documents.

MD04 Conformance Claim for Basic Profile 1.1 and Basic Security Profile 1.0 MUST be incorporated in WSDL documents [CCAM].

Note that SSEK is based on Basic Profile 1.1, which means that WSDL must be used according to Basic Profile 1.1 guidelines.

6.1 SSEK Policy Language

WS-Policy [WSPOLICY] [WSPOLICYATT] defines a framework for describing web service behaviours. Behaviour refers for example to whether messages must be signed or if SSL must be used. Policies can be used to define the following aspects of SSEK for a web service.

- Whether or not TxId will be used
- How senders and receivers must be identified in messages
- Whether or not resending is generally allowed
- Which asynchronous message flow types are supported
- Which transport security will be used
- Which message security will be used.
- How the query message signature will be returned to the sender

Simplified description of SSEK Policy Language (http://schemas.ssek.org/ssek/2006-05-10/policy). For an exact description see the appended schema.

```
<ssekp:ServiceAssertions>
  <ssekp:IdType>CN|DN|ORGNR|APP<ssekp:IdType>?
  <ssekp:UseTxId/>?
  <ssekp:TransportLevelSecurity>SSL|SSLWithClientCertificate</ssekp:TransportLevelSecurity003E
  <ssekp:MessageLevelSecurity>None|Signature</ssekp:MessageLevelSecurity>
  <ssekp:RequestSignatureHandling>
       Receipt|SignatureConfirmation|ReceiptAndSignatureConfirmation
  </ssekp:RequestSignatureHandling>?
<ssekp:ServiceAssertions>
<ssekp:OperationAssertions>
  <ssekp:ResendAllowedOnTimeout/>?
  <ssekp:SupportsAsynchPull/>?
  <ssekp:SupportsAsynchPush/>?
</ssekp:OperationAssertions>
```

/ssekp:ServiceAssertion/ssekp:IdType

The IdType element indicates which type of SenderId and ReceiverId must be stated for the web service. See the SSEK schema for a description of the contents.

/ssekp:ServiceAssertion/ssekp:UseTxId

The UseTxId elements indicates that TxId must be used when calling the web service.

/ssekp:ServiceAssertion/ssekp:TransportLevelSecurity

The TransportLevelSecurity element indicates the security that must be used at the transport level for the web service (either SSL or SSLWithClientCertificate).

/ssekp:ServiceAssertion/ssekp:MessageLevelSecurity

The MessageLevelSecurity element indicates the security that must be used at the message level for the web service (None or Signature).

/ssekp:ServiceAssertion/ssekp:RequestSignatureHandling

The RequestSignatureHandling element indicates how the signature in the query message must be returned to the sender (Receipt or SignatureConfirmation or ReceiptAndSignatureConfirmation). Receipt means that a standard receipt is returned with the query message signature. SignatureConfirmation means that the procedure defined in [WSS11] is used to return query message signatures and ReceiptAndSignatureConfirmation means that both methods are used.

/ssekp:OperationAssertion/ssekp:ResendAllowedOnTimeout

The ResendAllowedOnTimeout element indicates that resending of messages is allowed with no restrictions on the operation.

/ssekp:OperationAssertion/ssekp:SupportsAsynchPull

The SupportsAsynchPull element indicates that the operation can initiate an asynchronous message flow with retrieval.

/ssekp:OperationAssertion/ssekp:SupportsAsynchPush
The SupportsAsynchPush element indicates that the operation can initiate an asynchronous message flow with delivery.

According to the terminology presented in [WSPOLICYATT] the SSEK policy must be placed in the WSDL file for a web service as follows:

MD05 ssekp:ServiceAssertion MUST be referred to from wsdl:binding.

MD06 ssekp:OperationAssertion MUST be referred to from wsdl:binding./wsdl:operation.

7 SSEK 2.0 as a Component of Business Agreements

The SSEK:1.1 specification was developed to enable B2B communication of business-critical and sensitive information. The requirements for the specification were faced with a practical business challenge: communication of business information formerly executed on a signed paper document had to be replaced by electronic communication. The requirements were based on legal, security and commercial considerations. The information sent had to protected against unauthorised access and be securely linked to the sending party. SSEK 1.1 is now used for practical communication of sensitive, business-critical information, which confirms that the specification meets the set requirements.

SSEK 2.0 is an improved specification that provides even better opportunities for efficient electronic business communication through expanded functionality and modernisation of the standards on which SSEK 2.0 is based.

Using SSEK2.0 enables secure use of web services but in addition to specifying that SSEK 2.0 will be used, parties must agree on a number of items (in a business agreement for instance). The foundation of secure electronic business communication is agreement on these items and on the use of SSEK 2.0.

The items on which the parties need to agree may be divided into two areas: items that SSEK 2.0 specifies but for which options are provided in the specification and items that are outside the scope of the specification. In order to securely provide agreed web services published according to SSEK 2.0 in practical service utilisation, the following items SHOULD be agreed by the parties.

Items that have to be defined and for which options are given by SSEK 2.0:

- SenderId for the communicating parties
- Receiver Id for the communicating parties
- Producer URL.
- Consumer URL where applicable
- The levels of transport and message security to be used
- The revocation model to be used

Items that do not have to be defined and which are not specified by SSEK 2.0:

- The CA (Certificate Authority) approved for certificates intended for signature and authentication
- The type of certificate allowed for signature and authentication from the agreed CA
- The legal significance of communicated documents

8 Differences Compared to SSEK 1.1

Brief review of changes between SSEK 1.1 and SSEK 2.0:

- OASIS WS-Security is used for message signature: SOAP message Security 1.0/1.1
- Simplified processing of TxId
- Timestamp in TxHeader has been eliminated from the SSEK header and replaced by timestamp in WS-Security.
- Resending messages is allowed under certain conditions

- Standardised fault management
- All message structures in SSEK use the same namespace URI, http://schemas.ssek.org/ssek/2006-03-01/
- Support for asynchronous message flows with retrieval
- Support for Policies
- Harmonisation with WS-I Basic Profile 1.1 and Basic Security Profile 1.0

9 References

[BP11]	WS-I Basic Profile Version 1.1, 24 August 2004
[BP11ERR]	WS-I Basic Profile Version 1.1 Errata, Board Approval Draft, Rev: 1.8, 25 October 2005
[BSP10]	WS-I Basic Security Profile Version 1.0, Working Group Draft, 20 January 2006
[C14N]	Exclusive XML Canonicalization, Version 1.0, W3C Recommendation 18 July 2002 (http://www.w3.org/TR/2002/REC-xml-exc-c14n-20020718/)
[CCAM]	WS-I Conformance Claim Attachment Mechanism Version 1.0, Final Material 15 November 2004
[MIME]	MIME Media Types, Internet Assigned Numbers Authority
[MTOM]	SOAP Message Transmission Optimisation Mechanism, W3C Recommendation 25 January 2005
[OCSP]	Online Certificate Status Protocol, http://www.ietf.org/rfc/rfc2560.txt
[PKI]	Public-Key Infrastructure (X.509) (pkix), http://www.ietf.org/html.charters/pkix-charter.html
[SOAP]	Simple Object Access Protocol (SOAP) 1.1, W3C Note, 8 May 2000
[SOAP11MTOM]	SOAP 1.1 Binding for MTOM 1.0, 2 March 2006
[UUID]	RFC4122: A Universally Unique IDentifier (UUID) URN Namespace, http://www.ietf.org/rfc/rfc4122.txt
[WSA]	Web Services Addressing 1.0 - Core, W3C Candidate Recommendation 17 August 2005, http://www.w3.org/TR/2005/CR-ws-addr-core-20050817/
[WSASB]	Web Services Addressing 1.0 - SOAP Binding, W3C Candidate Recommendation 17 August 2005, http://www.w3.org/TR/2005/CR-ws-addr-soap-20050817/
[WSDL]	Web Services Description Language (WSDL) 1.1, W3C Note, 15 March 2001
[WSPOLICY]	Web Services Policy Framework (WS-Policy), September 2004
[WSPOLICYATT]	Web Services Policy Attachment (WS-PolicyAttachment), September 2004
[WSS10]	Web Services Security: SOAP Message Security 1.0, (WS-Security 2004), OASIS Standard 200401, March 2004
[WSS11]	Web Services Security: SOAP Message Security 1.1, (WS-Security 2004), OASIS Standard Specification, 1 February 2006
[X509CTP]	Web Services Security X.509 Certificate Token Profile 1.1, OASIS Standard Specification, 1 February 2006
[X509V3]	Internet X.509 Public Key Infrastructure, Certificate and CRL Profile, http://www.ietf.org/rfc/rfc2459.txt
[XML]	Extensible Markup Language (XML) 1.0 (Third Edition), W3C Recommendation 04 February 2004. http://www.w3.org/TR/2004/REC-xml-20040204/
[XMLBD]	Assigning Media Types to Binary Data in XML, W3C Working Draft 2 November 2004
[XMLDSIG]	XML-Signature Syntax and Processing, W3C Recommendation, 12 February 2002. http://www.w3.org/TR/xmldsig-core/.
[XMLSCHEMA1]	XML Schema Part 1: Structures Second Edition, W3C Recommendation 28 October 2004. http://www.w3.org/TR/xmlschema-1/
[XMLSCHEMA2]	XML Schema Part 2: Datatypes Second Edition, W3C Recommendation 28 October 2004. http://www.w3.org/TR/xmlschema-2/
[XOP]	XML-binary Optimized Packaging, W3C Recommendation 25 January 2005

Appendix 1: XML Schema for SSEK

```
<xsd:schema targetNamespace="http://schemas.ssek.org/ssek/2006-05-10/"
       elementFormDefault="qualified" attributeFormDefault="qualified"
       xmlns:tns="http://schemas.ssek.org/ssek/2006-05-10/"
       xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:simpleType name="IdType">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="APP"/>
      <xsd:enumeration value="CN"/>
      <xsd:enumeration value="DN"/>
      <xsd:enumeration value="ORGNR"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="FaultCode">
    <xsd:restriction base="xsd:QName">
      <xsd:enumeration value="AsynchMethodUnsupported"/>
      <xsd:enumeration value="ContentInvalid"/>
      <xsd:enumeration value="IncorrectContext"/>
      <xsd:enumeration value="MessageNotProcessed"/>
      <xsd:enumeration value="NoResultAvailable"/>
      <xsd:enumeration value="ReceiverIdUnknown"/>
      <xsd:enumeration value="SenderIdUnknown"/>
      <xsd:enumeration value="Timeout"/>
      <xsd:enumeration value="TxIdInvalid"/>
      <xsd:enumeration value="TxIdMissing"/>
      <xsd:enumeration value="TxIdNotAllowed"/>
      <xsd:enumeration value="TxIdUnknown"/>
      <xsd:enumeration value="WebServiceUnavailable"/>
      <xsd:enumeration value="WebServiceUnsupported"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:element name="SSEK">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="SenderId">
          <xsd:complexType>
            <xsd:simpleContent>
              <xsd:restriction base="xsd:anyType">
                <xsd:simpleType>
                  <xsd:restriction base="xsd:string">
                    <xsd:maxLength value="512"/>
                  </xsd:restriction>
                </xsd:simpleType>
                <xsd:attribute name="Type" default="CN" type="tns:IdType"/>
              </xsd:restriction>
            </xsd:simpleContent>
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="ReceiverId">
          <xsd:complexType>
            <xsd:simpleContent>
              <xsd:restriction base="xsd:anyType">
                <xsd:simpleType>
                  <xsd:restriction base="xsd:string">
                    <xsd:maxLength value="512"/>
                  </xsd:restriction>
                </xsd:simpleType>
                <xsd:attribute name="Type" default="CN" type="tns:IdType"/>
              </xsd:restriction>
            </xsd:simpleContent>
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="TxId" minOccurs="0">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:length value="36"/>
            </xsd:restriction>
```

```
          </xsd:simpleType>
        </xsd:element>
      </xsd:sequence>
      <xsd:attribute name="AsynchMethod" use="optional">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:enumeration value="AsynchPull"/>
            <xsd:enumeration value="AsynchPush"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
      <xsd:anyAttribute processContents="lax"/>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="FaultData">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="FaultingMessage" type="xsd:string" minOccurs="0"/>
        <xsd:element name="TxId" minOccurs="0">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:length value="36"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element minOccurs="0" name="FaultItems">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element maxOccurs="unbounded" name="FaultItem">
                <xsd:complexType>
                  <xsd:sequence>
                    <xsd:element name="Code" type="xsd:string"/>
                    <xsd:element name="Description" type="xsd:string" minOccurs="0"/>
                    <xsd:element name="Location" type="xsd:string" minOccurs="0"/>
                    <xsd:element name="System" type="xsd:string" minOccurs="0"/>
                  </xsd:sequence>
                </xsd:complexType>
              </xsd:element>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
        <xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded" namespace="##other"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Receipt">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="ResponseCode">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:enumeration value="OK"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element name="ResponseMessage" type="xsd:string" minOccurs="0"/>
        <xsd:element name="RequestSignatureValue" type="xsd:string" minOccurs="0"/>
        <xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded" namespace="##other"/>
      </xsd:sequence>
      <xsd:anyAttribute processContents="lax"/>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="PullMessage">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:anyAttribute processContents="lax"/>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Appendix 2: XML Schema for SSEK Policy Language

```
<xsd:schema targetNamespace="http://schemas.ssek.org/ssek/2006-05-10/policy"
       elementFormDefault="qualified" attributeFormDefault="qualified"
       xmlns:tns="http://schemas.ssek.org/ssek/2006-05-10/policy"
       xmlns:xsd="http://www.w3.org/2001/XMLSchema"
       xmlns:ssek="http://schemas.ssek.org/ssek/2006-05-10/">
  <xsd:element name="ServiceAssertion">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="IdType" type="ssek:IdType" minOccurs="0"/>
        <xsd:element minOccurs="0" name="UseTxId"/>
        <xsd:element name="TransportLevelSecurity">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:enumeration value="SSL"/>
              <xsd:enumeration value="SSLWithClientCertificate"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element name="MessageLevelSecurity">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:enumeration value="None"/>
              <xsd:enumeration value="Signature"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element name="RequestSignatureHandling" minOccurs="0">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:enumeration value="Receipt"/>
              <xsd:enumeration value="SignatureConfirmation"/>
              <xsd:enumeration value="ReceiptAndSignatureConfirmation"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="OperationAssertion">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element minOccurs="0" name="ResendAllowedOnTimeout"/>
        <xsd:element minOccurs="0" name="SupportsAsynchPull"/>
        <xsd:element minOccurs="0" name="SupportsAsynchPush"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```