LEGITIMATIONS NÄMNDEN

OSIF2 – Signature Service API

WSDL API Specification

Version 0.12
2013-05-21

www.elegnamnden.se

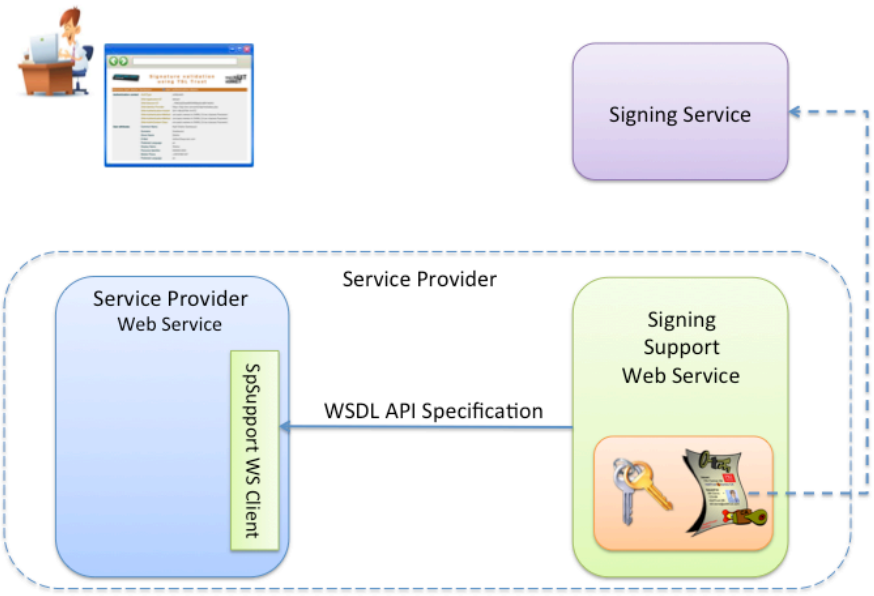
Postadress	Besöksadress	Telefon växel	E-postadress
171 94 SOLNA	Korta gatan 10	010-574 21 00	kansliet@elegnamnden.se

LEGITIMATIONS NÄMNDEN

1 API Overview

This document defines the operations and data types for the signature support web service API. This API is used to request services from a support service in accordance with the following schematic overviews:

Setup:

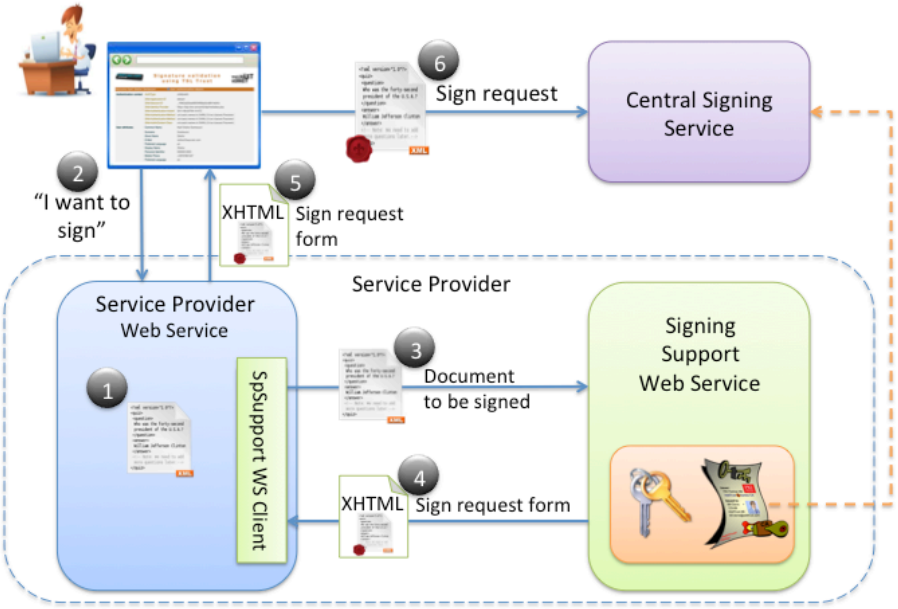


A service provider providing services to users that requires signing of electronic documents adds the capability to sign using a signing service by deploying a signing support web service. The support service exposes a web service API through a WSDL (Web Services Descriptive Language). This document specifies the API defined by this WSDL.

The support service generates necessary keys (or is configured with a signing key and a certificate). The certificate of the support service is handed to the signing service as a certificate representing the service provider as a legitimate requester of signing services.

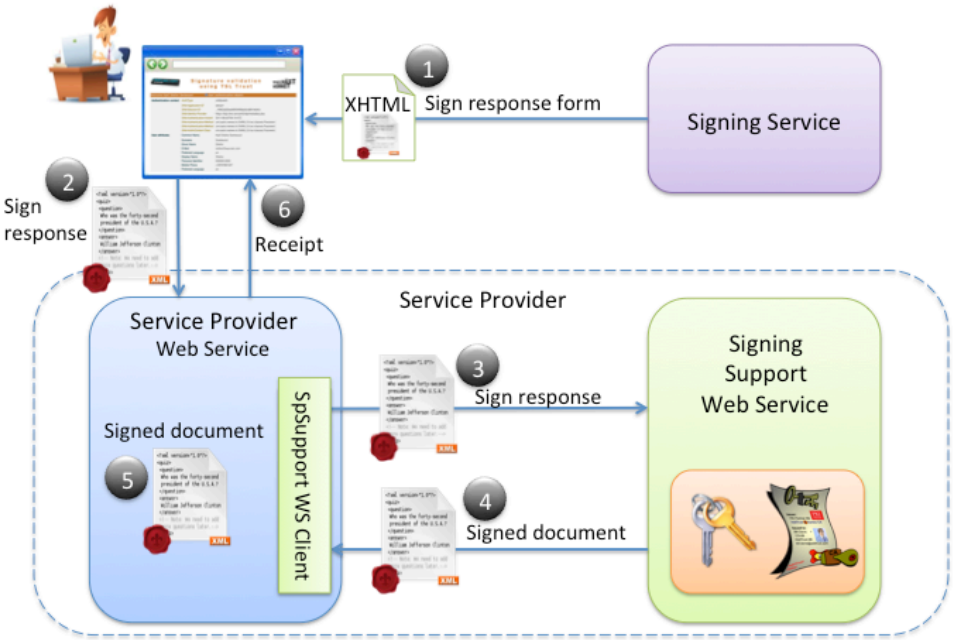
LEGITIMATIONS NÄMNDEN

Sign Request operation:



1. The Service provider produces or obtains some document that needs to be signed by the user.
2. The user agrees to sign.
3. The document to be signed is sent along with necessary parameters to a signing support service using the signRequest operation of the web services API.
4. A sign request XHTML page is returned to the service provider.
5. The XHTML page is returned to the user as a response to the agreement to sign.
6. The user's web browser renders the XHTML page, causing the browser to post a sign request to the signature service.

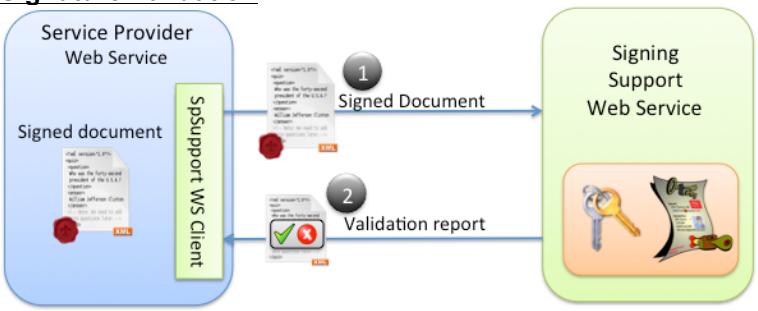
Complete Signing operation:



1. After completing the signing process, the signing service receives an XHTML page with sign response data from the signing service

2. The user's web browser renders the XHTML page, causing the browser to post the sign response to the service provider.
3. The sign response is sent using the present API to the support service using the completeSigning operation.
4. The support service assembles the complete signed document using data from the sign response and returns the signed document to the service provider.
5. The service provider processes the signed document to determine that the requested signature was successfully completed.
6. A receipt is sent to the user.

Signature Validation



1. A signed document is sent to the support service for validation using the signatureValidation operation of the present API.
2. The signed document is validated and a signature validation report is returned to the service provider.

1.1 Relationship with the legacy OSIF API

OSIF is a web service API used by service providers to integrate electronic services with the current national eID infrastructure.

The current OSIF protocol handles both authentication and signing.

In the new infrastructure for Swedish eID, authentication is handled using SAML assertions which is supported by various SAML integration products with their own APIs and methods for integration with web services. This new infrastructure requires a new signing service that can handle users that don't have their own signing key and signing certificate.

As there is no more need for any OSIF protocol for authentication services, but a need for a new corresponding API for integration with the new signing service, this API is proposed to replace the current OSIF protocol as service providers move to the new eID infrastructure.

1.2 Definitions

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

LEGITIMATIONS NÄMNDEN

2 Data Elements

Data elements are used as input and output data in operations provided by the API. Data elements are exchanged in SOAP messages as XML elements. The syntax of all data elements are specified through data types in the XML schema that is included in the WSDL description of the API.

The following sections specify the data content in defined data types.

2.1 Generic data types

Generic data elements are re-used in several elements used as input and output elements in operations of the API.

2.1.1 Status

Description
The status data type is used to carry status information for an operation, or part of an operation.

Schema

```
<xs:complexType name="status">
  <xs:sequence>
    <xs:element name="statusCode" type="xs:int"/>
    <xs:element name="statusCodeDescription" type="xs:string" minOccurs="0"/>
    <xs:element name="statusGroup" type="xs:int"/>
    <xs:element name="statusGroupDescription" type="xs:string" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

Element description

Name	Data type	Occurs	Description
statusGroup	int	[1..1]	Status group code
statusGroupDescription	string	[0..1]	Description of the status group
statusCode	int	[1..1]	Status code
statusCodeDescription	string	[0..1]	Description of the status code

Status codes and descriptions are defined in section 3.1.

2.2 Properties and parameters

Parameters in the input output data types of this API are specified either as a defined property or an extensible parameter using a key/value pair.

2.2.1 properties

Properties are key/value pairs where the key is an enumerated property. Only defined properties are allowed in the API. The list of defined properties may be expanded over time under the control of the Swedish E-identification board.

A property key is defined as an enumerated value of the Property data type defined under each input output data type of the API that use defined properties. In the current version of the API, only the SignRequestParams data types have defined properties.

2.2.2 parameters

parameters are key value pairs where the key is specified as a string. Properties provide protocol extensibility and a place to provide implementer specific parameters in addition to the defined parameters. Each data type that carries properties define their own rules for properties.

2.3 Operation elements

2.3.1 signRequestParams

Description
The signRequestParams data type is an input parameter to the signRequest operation.

Schema

```
<xs:complexType name="signRequestParams">
  <xs:sequence>
    <xs:element name="certType" type="tns:certType" minOccurs="0"/>
    <xs:element name="idpEntityId" type="xs:string" minOccurs="0"/>
    <xs:element name="loa" type="tns:signerAuthLoa" minOccurs="0"/>
    <xs:element name="parameters">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="entry" minOccurs="0" maxOccurs="unbounded">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="key" minOccurs="0" type="xs:string"/>
                <xs:element name="value" minOccurs="0" type="xs:string"/>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="properties">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="entry" minOccurs="0" maxOccurs="unbounded">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="key" minOccurs="0" type="tns:property"/>
                <xs:element name="value" minOccurs="0" type="xs:string"/>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="signTaskParams" type="tns:signTaskParams" nillable="true" minOccurs="0"
```

```
                maxOccurs="unbounded"/>
      <xs:element name="signerId" type="xs:string" minOccurs="0"/>
      <xs:element name="signerIdAttr" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
```

Element description

Name	Data type	Occurs	Description
signerId	xs:string	[1..1]	An identifier value of the signer. Normally a Swedish "personnummer"
signerIdAttr	xs:string	[1..1]	The attribute type defining the type of id value provided in signerId. This value is provided as the SAML attribute name URI.
idpEntityId	xs:string	[1..1]	The EntityId of the Identity Provider that will authenticate the signer at the signing service.
certType	certType	[0..1]	The type of signature certificate requested.
loa	signerAuthLoa	[0..1]	The requested minimum level of assurance used to authenticate the signer at the signing service.
properties	properties	[0..1]	Values of defined properties in accordance with 2.3.1.1. If no properties are defined, this element is absent. If one or more properties are defined, then one properties element is present with one entry for each defined property.
parameters	parameters	[0..1]	Optional set of parameter key/value pairs according to 2.2.2.
signTaskParams	signTaskParams	[1..n]	The signTaskParams element holds information about one requested signature. A signRequest MUST contain at least one instance of signTaskParams.

2.3.1.1 Defined properties

The following enumerated properties are defined for use in the SignRequestParams data type:

Value	Description
returnUrl	The URL to which the sign response from the signature service must be sent.
requestedAlgorithm	A URI specifying the requested signature algorithm
signMessage	A text/html message to be displayed to the user at the signature service. This sign message will be handled according to the operational policy of the signature service. By default, signature services do not display sign messages to users, even if such message is provided using this API.
spEntityId	THE EntityID of the service provider requesting a signature from the signature service.
requestedAttributes	A Base64 encoded XML element <eid2:RequestedCertAttributes> element as specified in [EID2-DSS]. The Base64 encoding SHALL be performed over a byte sequence representing the UTF-8 encoded characters of the XML element.

2.3.1.2 signTaskParams

Description

This data type provides information about one particular requested signature as part of a signRequestParams element.

Schema

```
<xs:complexType name="signTaskParams">
  <xs:sequence>
    <xs:element name="adesType" type="tns:adesType" minOccurs="0"/>
    <xs:element name="parameters">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="entry" minOccurs="0" maxOccurs="unbounded">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="key" minOccurs="0" type="xs:string"/>
                <xs:element name="value" minOccurs="0" type="xs:string"/>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="policy" type="xs:string" minOccurs="0"/>
    <xs:element name="sigType" type="tns:sigType" minOccurs="0"/>
    <xs:element name="signTaskId" type="xs:string" minOccurs="0"/>
    <xs:element name="tbsDocument" type="xs:base64Binary" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

Element description

Name	Data type	Occurs	Description
tbsDocument	xs:base64Binary	[1..1]	The bytes of the document to be signed
signTaskId	xs:string	[0..1]	An identifier of this signature task. This parameter MUST be present if more than one signTaskParams element is present and MUST in such case contain an ID that is unique among all signTaskParams in the signRequestParams element. This element is OPTIONAL if only one SignTaskParams element is present.
sigType	sigType	[0..1]	The type of signature requested according to 3.2.4.
adesType	adesType	[0..1]	The type of AdES profile requested according to 3.2.3.
policy	xs:string	[0..1]	An optional policy identifier, specifying any policy under which the signature must be created. This policy identifier MUST contain a URI identifier.
parameters	parameters	[0..1]	Optional set of parameter key/value pairs according to 2.2.2.

2.3.2 signRequestXhtml

Description

The signRequestXhtml data type is the output of the signRequest operation.

Schema

```
<xs:complexType name="signRequestXhtml">
  <xs:sequence>
    <xs:element name="signRequestXhtml" type="xs:base64Binary" minOccurs="0"/>
    <xs:element name="status" type="tns:status" minOccurs="0"/>
```

```
      <xs:element name="transactionId" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
```

Element description

Name	Data type	Occurs	Description
status	status	[1..1]	A status indication according to 2.1.1
transactionId	xs:string	[1..1]	An Id for this pending signature request. This id is used to map responses from completeSigning operations with the present signRequest operation.
signRequestXhtml	xs:base64Binary	[0..1]	A base64 encoded sequence of UTF-8 characters providing an XHTML page that will case the signer client to send a sign request to the signing server with a sign request for the requested signatures.

2.3.3 signResponse

Description

The signResponseParams element is input to the completeSigning operation.

Schema

```
<xs:element name="signResponse" type="xs:base64Binary" nillable="true" minOccurs="0"/>
```

Element description

Name	Data type	Occurs	Description
signResponse	xs:base64Binary	[1..1]	The bytes of a sign response from the signing service

2.3.4 signatureResult

Description

The signature result data type provides output from the completeSigning operation.

Schema

```
<xs:complexType name="signatureResult">
  <xs:sequence>
    <xs:element name="parameters">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="entry" minOccurs="0" maxOccurs="unbounded">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="key" minOccurs="0" type="xs:string"/>
                <xs:element name="value" minOccurs="0" type="xs:string"/>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="signTaskResult" type="tns:signTaskResult" nillable="true" minOccurs="0"
```

```
                maxOccurs="unbounded"/>
      <xs:element name="signerId">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="entry" minOccurs="0" maxOccurs="unbounded">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="key" minOccurs="0" type="xs:string"/>
                  <xs:element name="value" minOccurs="0" type="xs:string"/>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="status" type="tns:status" minOccurs="0"/>
      <xs:element name="transactionId" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
```

Element description

Name	Data type	Occurs	Description
status	xs: status	[1..1]	A status indication according to 2.1.1
transactionId	xs:string	[1..1]	The transaction Id identifying the signRequest that corresponds to the present signatureResult.
signerId	signerId	[1..1]	A sequence of entries holding key value pairs, where the key holds a name of an Id attribute of the signer and the value holds the corresponding attribute value. The name of the attribute is primary used for presentation purposes and MAY hold a suitable presentation name of the subject attribute. This element provides convenience for UI design. For secure mapping of users, the data provided in the sign response in combination with the signature certificate SHOULD be used.
parameters	parameters	[0..1]	Optional set of parameter key/value pairs according to 2.2.2.
signTaskResult	signTaskResult	[0..1]	Result data for each sign task as specified in 2.3.6.

2.3.5 signTaskResult

Description

The signTaskResult data type holds result information from each sign task in a signatureResult data type.

Schema

```
  <xs:complexType name="signTaskResult">
    <xs:sequence>
      <xs:element name="parameters">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="entry" minOccurs="0" maxOccurs="unbounded">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="key" minOccurs="0" type="xs:string"/>
                  <xs:element name="value" minOccurs="0" type="xs:string"/>
```

```
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="signTaskId" type="xs:string" minOccurs="0"/>
      <xs:element name="signedDoc" type="xs:base64Binary" minOccurs="0"/>
      <xs:element name="signedDocRef" type="xs:string" minOccurs="0"/>
      <xs:element name="status" type="tns:status" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
```

Element description

Name	Data type	Occurs	Description
status	xs: status	[1..1]	A status indication according to 2.1.1
signTaskId	xs:string	[1..1]	The signTaskId corresponding to the signTaskId provided in the signRequest operation for the sign task that provided input to the process that produced the present result.
signedDoc	xsbase64Binary	[0..1]	The bytes of the signed document.
signedDocRef	xs:string	[0..1]	A URL to a location where the signed document can be obtained. At least one of signedDoc or signedDocRef MUST be present.
parameters	parameters	[0..1]	Optional set of parameter key/value pairs according to 2.2.2.

2.3.6 verifySignature

Description

The verify signature data type provides input to the verifySignature operation.

Schema

```
  <xs:complexType name="verifySignature">
    <xs:sequence>
      <xs:element name="signedDocument" type="xs:base64Binary" nillable="true" minOccurs="0"/>
      <xs:element name="validationPolicy" type="xs:string" minOccurs="0"/>
      <xs:element name="parameters">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="entry" minOccurs="0" maxOccurs="unbounded">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="key" minOccurs="0" type="xs:string"/>
                  <xs:element name="value" minOccurs="0" type="xs:string"/>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
```

LEGITIMATIONS NÄMNDEN

Element description

Name	Data type	Occurs	Description
signedDocument	xs:base64Binary	[1..1]	The bytes of a signed document to be verified.
validationPolicy	xs:string	[0..1]	The name of a policy governing the valid set of trusted certificate issuers as well as other rules for signature verification.
parameters	parameters	[0..1]	Optional set of parameter key/value pairs according to 2.2.2.

2.3.7 verifyResponse

Description

The verify signature data type provides input to the verifySignature operation.

Schema

```
<xs:complexType name="verifyResponse">
  <xs:sequence>
    <xs:element name="status" type="tns:status" minOccurs="0"/>
    <xs:element name="verifyReport" type="xs:base64Binary" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

Element description

Name	Data type	Occurs	Description
status	xs:string	[1..1]	A status indication according to 2.1.1
verifyReport	xs:base64Binary	[0..1]	The bytes of a status report according to the validation service used to validate the signature.

LEGITIMATIONS NÄMNDEN

3 Parameters and constants

3.1 Status

The following status codes provide a preliminary starting point for this API. These status codes are likely to change or be updated.

3.1.1 Status groups

The following status groups are defined:

Name	Code	Description
Generic	1	Generic
SigRequest	2	Request for Signature
SignedDocCompletion	3	Completion of signed Document
SignatureValidation	4	Signature validation

3.1.2 Status codes

The following status codes are defined:

Name	Code	Description
OK	101	Operation successful
NoSignTask	201	No signing task in request
IllegalDocType	202	Illegal document type
ReqGenError	203	Failed to generate sign request
FailedSigCompletion	301	Failed to complete signing process
FailedValidation	401	Signature Validation Failed
NoValidationPolicy	402	No signature validation policy

3.2 Enumerations

The following enumerated values are used in data types of the API. The enumerated values of these data types can be expanded over time under the control of the Swedish E-identification Board.

3.2.1 CertType

CertType specifies a type of certificate. The following enumerated values are defined:

Value	Description
PKC	An X509 certificate that is not issued as a Qualified Certificate.
QC	A certificate issued as a Qualified Certificate in accordance with EN 319 412-5. This certificate is Not associated with a Secure Signature Creation Device (SSCD) according to EN 319 412-5.
QC_SSCD	A certificate issued as a Qualified Certificate associated with a Secure Signature Creation Device (SSCD) in accordance with EN 319 412-5.

3.2.2 SignerAuthLoa

SignerAuthLoa holds an enumerated identifier of a defined level of assurance. The following enumerated values are defined:

Value	Description
loa1	The level of assurance level 1, identified by the context class reference: http://id.elegnamnden.se/loa/1.0/loa1
loa2	The level of assurance level 2, identified by the context class reference: http://id.elegnamnden.se/loa/1.0/loa2
loa3	The level of assurance level 3, identified by the context class reference: http://id.elegnamnden.se/loa/1.0/loa3
loa4	The level of assurance level 4, identified by the context class reference: http://id.elegnamnden.se/loa/1.0/loa4

3.2.3 AdesType

AdesType specifies a type of EU Advanced Electronic Signature (AdES) format according to the signature standard applicable to the applicable document type. For XML documents the applicable AdES standard is XAdES (ETSI TS 101 903 V1.4.2 ,2010-12). For PDF documents the applicable AdES standard is PAdES part 1-5 (ETSI TS 102 778). The following enumerated values are defined:

Value	Description
None	No AdES profile.
BES	The AdES-BES profile (Basic Electronic Signature)
EPES	The AdES-EPES profile (Explicit policy based Electronic Signature)

3.2.4 SigType

SigType specifies the a basic type of electronic signature. The following enumerated values are defined:

Value	Description
XML	XML Signature
PDF	PDF Signature
XFA	XML signature on XFA content in a PDF document
PDFandXFA	PDF signature and XML signature on XML content in the PDF document

LEGITIMATIONS NÄMNDEN

4 Operations

This section describes the operations of the API

4.1 signRequest

Description
Generates a sign request XHTML from a document to be signed together with information necessary to generate the request.

Schema

```
<operation name="signRequest">
   <input wsam:Action="http://csspsupport.sigserv.aaasec.com/SpSupportWs/signRequestRequest"
          message="tns:signRequest"/>
   <output wsam:Action="http://csspsupport.sigserv.aaasec.com/SpSupportWs/signRequestResponse"
           message="tns:signRequestResponse"/>
</operation>
```

Parameters

Name	Description
signRequest	Input data. This message holds an element of dataType signRequestParams. See section 2.3.1

Returns:
Element of data type signRequestXhtml. See section 2.3.2.

4.2 completeSigning

Description
Completes construction of a signed document based on a previous sign request and a sign response message from the signing service.

Schema

```
<operation name="completeSigning">
   <input wsam:Action="http://csspsupport.sigserv.aaasec.com/SpSupportWs/completeSigningRequest"
          message="tns:completeSigning"/>
   <output wsam:Action="http://csspsupport.sigserv.aaasec.com/SpSupportWs/completeSigningResponse"
           message="tns:completeSigningResponse"/>
</operation>
```

Parameters

Name	Description
signResponse	Input data. The bytes of a sign response from the signing service. See section 2.3.3.

Returns:
Element of data type signatureResult. See section 2.3.4.

LEGITIMATIONS NÄMNDEN

4.3 verifySignature

Description
Verifies signatures and timestamps on a signed document.

Schema

```
<operation name="verifySignature">
   <input wsam:Action="http://csspsupport.sigserv.aaasec.com/SpSupportWs/verifySignatureRequest"
          message="tns:verifySignature"/>
   <output wsam:Action="http://csspsupport.sigserv.aaasec.com/SpSupportWs/verifySignatureResponse"
           message="tns:verifySignatureResponse"/>
</operation>
```

Parameters

Name	Description
verifySignature	Input data. Signed documents and parameters. See section 2.3.6

Returns:
Element of data type verifyResponse. See section 2.3.7.

LEGITIMATIONS NÄMNDEN

5 WSDL

This section provides the content of the WSDL defining this API.

5.1 Main

The following provides the main WSDL definitions. The externally provided XML Schema is provided in section 5.2.

Note that the service element of this WSDL must be updated to provide an accurate address element pointing to the deployed service.

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
xmlns:wsp="http://www.w3.org/ns/ws-policy" xmlns:wsp1_2="http://schemas.xmlsoap.org/ws/2004/09/policy"
xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tns="http://csspsupport.sigserv.aaasec.com/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://schemas.xmlsoap.org/wsdl/" targetNamespace="http://csspsupport.sigserv.aaasec.com/"
name="SpSupportWs">
  <types>
    <xsd:schema>
      <xsd:import namespace="http://csspsupport.sigserv.aaasec.com/" schemaLocation="CsSpSupportWs.xsd"/>
    </xsd:schema>
  </types>
  <message name="verifySignature">
    <part name="parameters" element="tns:verifySignature"/>
  </message>
  <message name="verifySignatureResponse">
    <part name="parameters" element="tns:verifySignatureResponse"/>
  </message>
  <message name="signRequest">
    <part name="parameters" element="tns:signRequest"/>
  </message>
  <message name="signRequestResponse">
    <part name="parameters" element="tns:signRequestResponse"/>
  </message>
  <message name="completeSigning">
    <part name="parameters" element="tns:completeSigning"/>
  </message>
  <message name="completeSigningResponse">
    <part name="parameters" element="tns:completeSigningResponse"/>
  </message>
  <portType name="SpSupportWs">
    <operation name="verifySignature">
      <input wsam:Action="http://csspsupport.sigserv.aaasec.com/SpSupportWs/verifySignatureRequest"
            message="tns:verifySignature"/>
      <output wsam:Action="http://csspsupport.sigserv.aaasec.com/SpSupportWs/verifySignatureResponse"
             message="tns:verifySignatureResponse"/>
    </operation>
    <operation name="signRequest">
      <input wsam:Action="http://csspsupport.sigserv.aaasec.com/SpSupportWs/signRequestRequest"
            message="tns:signRequest"/>
      <output wsam:Action="http://csspsupport.sigserv.aaasec.com/SpSupportWs/signRequestResponse"
```

```
                message="tns:signRequestResponse"/>
    </operation>
    <operation name="completeSigning">
      <input wsam:Action="http://csspsupport.sigserv.aaasec.com/SpSupportWs/completeSigningRequest"
             message="tns:completeSigning"/>
      <output wsam:Action="http://csspsupport.sigserv.aaasec.com/SpSupportWs/completeSigningResponse"
              message="tns:completeSigningResponse"/>
    </operation>
  </portType>
  <binding name="SpSupportWsPortBinding" type="tns:SpSupportWs">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
    <operation name="verifySignature">
      <soap:operation soapAction=""/>
      <input>
        <soap:body use="literal"/>
      </input>
      <output>
        <soap:body use="literal"/>
      </output>
    </operation>
    <operation name="signRequest">
      <soap:operation soapAction=""/>
      <input>
        <soap:body use="literal"/>
      </input>
      <output>
        <soap:body use="literal"/>
      </output>
    </operation>
    <operation name="completeSigning">
      <soap:operation soapAction=""/>
      <input>
        <soap:body use="literal"/>
      </input>
      <output>
        <soap:body use="literal"/>
      </output>
    </operation>
  </binding>
  <service name="SpSupportWs">
    <port name="SpSupportWsPort" binding="tns:SpSupportWsPortBinding">
      <soap:address location="http://localhost:8080/CsSpSupport/SpSupportWs"/>
    </port>
  </service>
</definitions>
```

5.2 XML Schema

The CsSpSupportWs.xsd schema referenced in the WSDL description outlined in the previous section.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:tns="http://csspsupport.sigserv.aaasec.com/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
version="1.0" targetNamespace="http://csspsupport.sigserv.aaasec.com/">
  <xs:element name="completeSigning" type="tns:completeSigning"/>
```

```
<xs:element name="completeSigningResponse" type="tns:completeSigningResponse"/>
<xs:element name="signRequest" type="tns:signRequest"/>
<xs:element name="signRequestResponse" type="tns:signRequestResponse"/>
<xs:element name="verifySignature" type="tns:verifySignature"/>
<xs:element name="verifySignatureResponse" type="tns:verifySignatureResponse"/>
<xs:complexType name="verifySignature">
  <xs:sequence>
    <xs:element name="signedDocument" type="xs:base64Binary" nillable="true" minOccurs="0"/>
    <xs:element name="validationPolicy" type="xs:string" minOccurs="0"/>
    <xs:element name="parameters">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="entry" minOccurs="0" maxOccurs="unbounded">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="key" minOccurs="0" type="xs:string"/>
                <xs:element name="value" minOccurs="0" type="xs:string"/>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="verifySignatureResponse">
  <xs:sequence>
    <xs:element name="return" type="tns:verifyResponse" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="verifyResponse">
  <xs:sequence>
    <xs:element name="status" type="tns:status" minOccurs="0"/>
    <xs:element name="verifyReport" type="xs:base64Binary" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="status">
  <xs:sequence>
    <xs:element name="statusCode" type="xs:int"/>
    <xs:element name="statusCodeDescription" type="xs:string" minOccurs="0"/>
    <xs:element name="statusGroup" type="xs:int"/>
    <xs:element name="statusGroupDescription" type="xs:string" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="completeSigning">
  <xs:sequence>
    <xs:element name="signResponse" type="xs:base64Binary" nillable="true" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="completeSigningResponse">
  <xs:sequence>
    <xs:element name="return" type="tns:signatureResult" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="signatureResult">
```

```
    <xs:sequence>
      <xs:element name="parameters">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="entry" minOccurs="0" maxOccurs="unbounded">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="key" minOccurs="0" type="xs:string"/>
                  <xs:element name="value" minOccurs="0" type="xs:string"/>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="signTaskResult" type="tns:signTaskResult" nillable="true" minOccurs="0"
                  maxOccurs="unbounded"/>
      <xs:element name="signerId">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="entry" minOccurs="0" maxOccurs="unbounded">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="key" minOccurs="0" type="xs:string"/>
                  <xs:element name="value" minOccurs="0" type="xs:string"/>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="status" type="tns:status" minOccurs="0"/>
      <xs:element name="transactionId" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="signTaskResult">
    <xs:sequence>
      <xs:element name="parameters">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="entry" minOccurs="0" maxOccurs="unbounded">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="key" minOccurs="0" type="xs:string"/>
                  <xs:element name="value" minOccurs="0" type="xs:string"/>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="signTaskId" type="xs:string" minOccurs="0"/>
      <xs:element name="signedDoc" type="xs:base64Binary" minOccurs="0"/>
      <xs:element name="signedDocRef" type="xs:string" minOccurs="0"/>
      <xs:element name="status" type="tns:status" minOccurs="0"/>
```

```
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="signRequest">
    <xs:sequence>
      <xs:element name="signRequestParams" type="tns:signRequestParams" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="signRequestParams">
    <xs:sequence>
      <xs:element name="certType" type="tns:certType" minOccurs="0"/>
      <xs:element name="idpEntityId" type="xs:string" minOccurs="0"/>
      <xs:element name="loa" type="tns:signerAuthLoa" minOccurs="0"/>
      <xs:element name="parameters">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="entry" minOccurs="0" maxOccurs="unbounded">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="key" minOccurs="0" type="xs:string"/>
                  <xs:element name="value" minOccurs="0" type="xs:string"/>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="properties">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="entry" minOccurs="0" maxOccurs="unbounded">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="key" minOccurs="0" type="tns:property"/>
                  <xs:element name="value" minOccurs="0" type="xs:string"/>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="signTaskParams" type="tns:signTaskParams" nillable="true" minOccurs="0"
                  maxOccurs="unbounded"/>
      <xs:element name="signerId" type="xs:string" minOccurs="0"/>
      <xs:element name="signerIdAttr" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="signTaskParams">
    <xs:sequence>
      <xs:element name="adesType" type="tns:adesType" minOccurs="0"/>
      <xs:element name="parameters">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="entry" minOccurs="0" maxOccurs="unbounded">
              <xs:complexType>
                <xs:sequence>
```

```
                        <xs:element name="key" minOccurs="0" type="xs:string"/>
                        <xs:element name="value" minOccurs="0" type="xs:string"/>
                      </xs:sequence>
                    </xs:complexType>
                  </xs:element>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
            <xs:element name="policy" type="xs:string" minOccurs="0"/>
            <xs:element name="sigType" type="tns:sigType" minOccurs="0"/>
            <xs:element name="signTaskId" type="xs:string" minOccurs="0"/>
            <xs:element name="tbsDocument" type="xs:base64Binary" minOccurs="0"/>
          </xs:sequence>
        </xs:complexType>
        <xs:complexType name="signRequestResponse">
          <xs:sequence>
            <xs:element name="return" type="tns:signRequestXhtml" minOccurs="0"/>
          </xs:sequence>
        </xs:complexType>
        <xs:complexType name="signRequestXhtml">
          <xs:sequence>
            <xs:element name="signRequestXhtml" type="xs:base64Binary" minOccurs="0"/>
            <xs:element name="status" type="tns:status" minOccurs="0"/>
            <xs:element name="transactionId" type="xs:string" minOccurs="0"/>
          </xs:sequence>
        </xs:complexType>
        <xs:simpleType name="certType">
          <xs:restriction base="xs:string">
            <xs:enumeration value="PKC"/>
            <xs:enumeration value="QC"/>
            <xs:enumeration value="QC_SSCD"/>
          </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="signerAuthLoa">
          <xs:restriction base="xs:string">
            <xs:enumeration value="loa1"/>
            <xs:enumeration value="loa2"/>
            <xs:enumeration value="loa3"/>
            <xs:enumeration value="loa4"/>
          </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="property">
          <xs:restriction base="xs:string">
            <xs:enumeration value="returnUrl"/>
            <xs:enumeration value="requestedAlgorithm"/>
            <xs:enumeration value="signMessage"/>
            <xs:enumeration value="spEntityId"/>
            <xs:enumeration value="requestedAttributes"/>
          </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="adesType">
          <xs:restriction base="xs:string">
            <xs:enumeration value="None"/>
            <xs:enumeration value="BES"/>
            <xs:enumeration value="EPES"/>
```

```
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="sigType">
      <xs:restriction base="xs:string">
        <xs:enumeration value="XML"/>
        <xs:enumeration value="PDF"/>
        <xs:enumeration value="XFA"/>
        <xs:enumeration value="PDFandXFA"/>
      </xs:restriction>
    </xs:simpleType>
</xs:schema>
```

6 Appendix A – Revision History

Rev	Date	Update	Author
0.11	2013-05-20	First draft with revision history.	SS
0.12	2013-05-21	Corrections. Added operation descriptions	SS