DSS Extension for Federated Central Signing Services - Version 1.1

Swedish eID Framework Specification - 14 August 2015

Specification URIs:

This Version:

- http://aaa-sec.com/csig/specification/csig-dss-extension-1.1.html (Authoritative)\
- http://aaa-sec.com/csig/specification/csig-dss-extension-1.1.pdf
- http://aaa-sec.com/csig/specification/csig-dss-extension-1.1.docx

Previous Version:

- http://aaa-sec.com/eid2/specification/eid2dssext-1.0.html
- http://aaa-sec.com/eid2/specification/eid2dssext-1.0.pdf
- http://aaa-sec.com/eid2/specification/eid2dssext-1.0.xml

Latest Version:

- http://aaa-sec.com/eid2/specification/csig-dss-extension-1.1.html (Authoritative)

Technical Committee:

> The Swedish E-identification Board

Editor:

> Stefan Santesson, E-legitimationsnämnden <stefan@aaa-sec.com>

Related Work:

> Digital Signature Service Core Protocols, Elements, and Bindings Version 1.0 [DSS]

Abstract:

> This specification defines an esxtension to the OASIS DSS protocol for providing centralized signing services for E-government services within the identity federation for public agencies operated by the Swedish E-identification board.

Notices

This specification is defines an extension to the OASIS DSS protocol. It is provided in OASIS document format in order to allow it to be submitted to OASIS for adoption at a later stage. In it's current version, this specification is developed outside of OASIS technical committees.

Table of Contents

1. **Introduction**
   1.1. Terminology
   1.1.1. Key words
   1.1.2. Structure
   1.1.3. Definitions
   1.2. Normative References
   1.3. Non-Normative References
   1.4. Schema Organization and Namespaces
   1.5. Common Data Types
   1.5.1. String values
   1.5.2. URI Values
   1.5.3. Time Values
   1.5.4. MIME Types for <dss:TransformedData>
2. **Common Protocol Structures**
   2.1. Type AnyType
3. **Federated signing DSS Extensions**
   3.1. Element <SignRequestExtension>`
   3.1.1. Type CertRequestPropertiesType
   3.1.1.1. Type RequestedAttributesType
   3.1.2. Element <SignMessage>
   3.2. Element <SignResponseExtension>
   3.2.1. Type SignerAssertionInfoType
   3.2.1.1. Type ContextInfoType
   3.2.1.2. Type SAMLAssertionsType
   3.2.2. Type CertificateChainType
4. **Extensions to <dss:InputDocuments> and <dss:SignatureObject>**
   4.1. Element <SignTasks>
   4.1.1. Element <SignTaskData>
   4.1.1.1. Type AdESObjectType
5. **Signing sign requests and responses**

Appendixes

A. XML Schema

1. Introduction

This specifications defines elements that extends the <dss:SignRequest> and `<dss:SignResponse>` elements of [*OASIS-DSS*].

One element `<SignRequestExtension>` is defined for extending sign requests and one element `<SignResponseExtension>` is defined for extending sign responses. The `<SignTasks>` element extends the extends `<dss:InputDocuments>` of sign requests and `<dss:SignatureObject>` of sign responses.

These extensions to the DSS protocols provide essential protocol elements in a scenario where:

- The user's signature is requested by a service with which the user has an active session and where this service has authenticated the user using a SAML assertion
- The service provider holds the data to be signed
- The central signing service is requested to authenticate the user with the same *Identity Provider* that was used to authenticate the user to the Service provider.

This particular use case is relevant when a user logs in to a service using federated identity, where the user at some point is required to sign some data such as a tax declaration or payment transaction. The user is forwarded to the signing service together with a sign request from the service provider, specifying the information to be signed together with the conditions for signing. After completed signing, the user is returned to the service provider together with a sign response from which the service provider can assemble a complete signed document, signed by the user.

This scenario requires more information in both sign request and sign response than the original DSS core standard provides and it also requires the capability to let the service provider sign the sign request.

1.1. Terminology

1.1.1. Key words

The key words *MUST*, *MUST NOT*, *REQUIRED*, *SHALL*, *SHALL NOT*, *SHOULD*, *SHOULD NOT*, *RECOMMENDED*, *MAY*, and *OPTIONAL* are to be interpreted as described in [*RFC 2119*].

These keywords are capitalized when used to unambiguously specify requirements over protocol features and behavior that affect the interoperability and security of implementations.

When these words are not capitalized, they are meant in their natural-language sense.

1.1.2. Structure

This specification uses the following typographical conventions in text: `<ThisSpecifictionElements>`, `<ns:ForeignElement>`, Attribute, **Datatype**, OtherCode.

Listings of DSS schemas appear like this.

1.1.3. Definitions

Identity Provider

> An Identity Provider that is assigned to authenticate the signer

Signing Service

> A Signing Service that receives sign requests and responds with sign responses according to this specification.

Service Provider

> A service that has identified the signer through the user's Identity Provider and requests that the signer signs some data using the Central Signing Service.

1.2. Normative References

[Csig-XSD] This specification's DSS Extensions schema (Provided in this document), August 2015.

[OASIS-DSS] Digital Signature Service Core Protocols, Elements, and Bindings Version 1.0, *http://docs.oasis-open.org/dss/v1.0/oasis-dss-core-spec-v1.0-os.html*, OASIS, 11 April 2007.

[RFC 2119] S. Bradner, Key words for use in RFCs to Indicate Requirement Levels, *http://www.ietf.org/rfc/rfc2119.txt*, IETF (Internet Engineering Task Force) RFC 2119, March 1997.

[RFC 3986] T. Berners-Lee, R. Fielding, L. Masinter Uniform Resource Identifier (URI): Generic Syntax, *http://www.ietf.org/rfc/rfc3986.txt*, IETF (Internet Engineering Task Force) RFC 3986, January 2005.

[RFC 5652] R. Housley Cryptographic Message Syntax (CMS), *http://www.ietf.org/rfc/rfc5652.txt*, IETF (Internet Engineering Task Force) RFC 5652, September 2009.

[RFC 5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, *http://www.ietf.org/rfc/rfc5280.txt*, IETF (Internet Engineering Task Force) RFC 5280, May 2008.

[SAML2.0] Scott Cantor, John Kemp, Rob Philpott, Eve Maler Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0, *http://docs.oasis-open.org/security/saml/v2.0/*, OASIS Standard, 15 March 2005.

[Schema1] H. S. Thompson et al. XML Schema Part 1: Structures, *http://www.w3.org/TR/xmlschema-1/*, W3C Recommendation, May 2001.

[Schema2] P. V. Biron et al. XML Schema Part 2: Datatypes. *http://www.w3.org/TR/xmlschema-2/* , W3C Recommendation, May 2001.

[XAdES] XML Advanced Electronic Signatures, *http://www.etsi.org/deliver/etsi_ts/101900_101999/101903/01.04.02_60/ts_101903v010402p.pdf*, ETSI, December 2010.

[XML] Extensible Markup Language (XML) 1.0 (Fifth Edition), *http://www.w3.org/TR/REC-xml/\#sec-element-content*, W3C Recommendation 26 November 2008.

[XML-ns] T. Bray, D. Hollander, A. Layman. Namespaces in XML, *http://www.w3.org/TR/1999/REC-xml-names-19990114*, W3C Recommendation, January 1999.

[XMLDSIG] D. Eastlake et al, XML-Signature Syntax and Processing, *http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/*, W3C Recommendation, February 2002.

1.3. Non-Normative References

[W3C-CHAR] M. J. Dürst. Requirements for String Identity Matching and String Indexing. *http://www.w3.org/TR/WD-charreq/* , World Wide Web Consortium, July 1998.

1.4. Schema Organization and Namespaces

The structures described in this specification are contained in the XML schema for this protocol [*Csig-XSD*]. All schema listings in the current document are excerpts from the complete XML schema. In the case of a disagreement between the schema file and this document, the complete schema provided in the end of this document takes precedence.

This schema is associated with the following XML namespace:

http://id.elegnamnden.se/csig/1.1/dss-ext/ns

If a future version of this specification is needed, it will use a different namespace.

Conventional XML namespace prefixes are used in the schema:

- The prefix `csig`: stands for the this specification's XML schema namespace [*Csig-XSD*].
- The prefix `dss`: stands for the DSS core namespace [*OASIS-DSS*].
- The prefix `ds`: stands for the W3C XML Signature namespace [*XMLDSIG*].
- The prefix `xs`: stands for the W3C XML Schema namespace [*Schema1*].
- The prefix `saml`: stands for the OASIS SAML Schema namespace [*SAML2.0*].
- The prefix `saml`: stands for the ETSI XAdES Schema namespace [*XAdES*].

Applications MAY use different namespace prefixes, and MAY use whatever namespace defaulting/scoping conventions they desire, as long as they are compliant with the Namespaces in XML specification [*XML-ns*].

The following schema fragment defines the XML namespaces and other header information for this specification's core XML schema:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified"
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
    targetNamespace="http://id.elegnamnden.se/csig/1.1/dss-ext/ns"
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:dss="urn:oasis:names:tc:dss:1.0:core:schema"
    xmlns:csig="http://id.elegnamnden.se/csig/1.1/dss-ext/ns">
    <xs:import namespace="urn:oasis:names:tc:SAML:2.0:assertion"
        schemaLocation="saml-schema-assertion-2.0.xsd"/>
```

1.5. Common Data Types

The following sections define how to use and interpret common data types that appear throughout the specification.

1.5.1. String values

All string values have the type **xs:string**, which is built in to the W3C XML Schema Datatypes specification [*Schema2*]. Unless otherwise noted in this specification or particular profiles, all strings in this specification MUST consist of at least one non-whitespace character (whitespace is defined in the XML Recommendation [*XML*] Section 2.3).

Unless otherwise noted in this specification, all elements that have the XML Schema **xs:string** type, or a type derived from that, MUST be compared using an exact binary comparison. In particular, implementations MUST NOT depend on case insensitive string comparisons, normalization or trimming of whitespace, or conversion of locale-specific formats such as numbers or

currency. This requirement is intended to conform to the W3C working-draft Requirements for String Identity, Matching, and String Indexing [*W3C-CHAR*].

1.5.2. URI Values

All SAML URI reference values have the type **xs:anyURI**, which is built in to the W3C XML Schema Datatypes specification [*Schema2*].

Unless otherwise indicated in this specification, all URI reference values used within defined elements or attributes MUST consist of at least one non-whitespace character, and are REQUIRED to be absolute [*RFC 3986*].

Note that this specification makes use of URI references as identifiers, such as status codes, format types, attribute and system entity names, etc. In such cases, it is essential that the values be both unique and consistent, such that the same URI is never used at different times to represent different underlying information.

1.5.3. Time Values

All time values have the type **xs:dateTime**, which is built in to the W3C XML Schema Datatypes specification [*Schema2*], and MUST be expressed in UTC form, with no time zone component.

Implementations SHOULD NOT rely on time resolution finer than milliseconds. Implementations MUST NOT generate time instants that specify leap seconds.

1.5.4. MIME Types for <dss:TransformedData>

The following MIME type is defined for transformed data included in a `<dss:Base64Data>` element within a `<dss:TransformedData>` element in a `<dss:SignRequest>`.

application/cms-signed-attributes

> This MIME type identifies that the data contained in the `<dss:Base64Data>` element, is a DER encoded CMS signed attributes structure [*RFC 5652*]. This data is useful when signing a PDF document in cases where the whole PDF document is not included in the request. The signature on a PDF document is generated by signing the hash of a CMS signed attributes structure representing the PDF document to be signed. These signed attributes includes data that a Signing Service may want to check before signing, such as the claimed signing time.

2. Common Protocol Structures

The following sections describe XML structures and types that are used in multiple places.

2.1. Type AnyType

The **AnyType** complex type allows arbitrary XML element content within an element of this type (see section 3.2.1 Element Content [*XML*]).

```
<xs:complexType name="AnyType">
    <xs:sequence>
        <xs:any processContents="lax" minOccurs="0"
              maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
```

3. Federated signing DSS Extensions

This section defines elements that extends the `<dss:SignRequest>` and `<dss:SignResponse>` elements of the DSS Signing Protocol.

3.1. Element <SignRequestExtension>`

The `<SignRequestExtension>` element allows a requesting service to add essential sign request information to a DSS Sign request. When present, this element MUST be included in the `<dss:OptionalInputs>` element in a DSS Sign Request. This element's **SignRequestExtensionType** complex type includes the following attributes and elements:

`Version` [Optional] (Default "`1.1`")

> The version of this specification. If absent, the version value defaults to "1.1". This attribute provides means for the receiving service to determine the expected syntax of the request based on the protocol version.

`<RequestTime>` [Required]

> The time when this request was created.

`<saml:Conditions>` [Required]

> Conditions that MUST be evaluated when assessing the validity of and/or when using the Sign Request. See Section 2.5 of [*SAML2.0*]for additional information on how to evaluate conditions.
>
> This element MUST include the attributes NotBefore and NotOnOrAfter and MUST include the element `<saml:AudienceRestriction>` which in turn MUST contain one `<saml:Audience>` element, specifying the return URL for any resulting Sign Response message.

`<Signer>` [Optional]

> The identity of the signer expressed as a sequence of SAML attributes using the **saml:AttributeStatementType** complex type. If this element is present, then the Signing Service MUST verify that the authenticated identity of the signer is consistent with the attributes in this element.

`<IdentityProvider>` [Required]

> The SAML EntityID of the Identity Provider that MUST be used to authenticate the signer before signing. The EntitID value is specified using the **saml:NameIDType** complex type and MUST include a `Format` attribute with the value `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`.

`<SignRequester>` [Required]

> The SAML EntityID of the service that sends this request to the Signing Service. The EntityID value is specified using the **saml:NameIDType** complex type and MUST include a `Format` attribute with the value `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`.

`<SignService>` [Required]

> The SAML EntityID of the service to which this Sign Request is sent. The EntityID value is specified using the **saml:NameIDType** complex type and MUST include a `Format` attribute with the value `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`.

`<RequestedSignatureAlgorithm>` [Optional]

> An identifier of the signature algorithm the requesting service prefers when generating the requested signature.

`<SignMessage>` [Optional]

> Optional sign message with information to the signer about the requested signature.

`<CertRequestProperties>` [Optional]

> An optional set of requested properties of the signature certificate that is generated as part of the signature process.

`<OtherRequestInfo>` [Optional]

> Any additional inputs to the request extension.

The following schema fragment defines the `<SignRequestExtension>` element and its **SignRequestExtensionType** complex type:

```
<xs:element name="SignRequestExtension" type="csig:SignRequestExtensionType"/>
<xs:complexType name="SignRequestExtensionType">
    <xs:sequence>
        <xs:element ref="csig:RequestTime"/>
        <xs:element ref="saml:Conditions"/>
        <xs:element ref="csig:Signer" minOccurs="0"/>
        <xs:element ref="csig:IdentityProvider"/>
        <xs:element ref="csig:SignRequester"/>
        <xs:element ref="csig:SignService"/>
        <xs:element minOccurs="0" ref="csig:RequestedSignatureAlgorithm"/>
        <xs:element minOccurs="0" ref="csig:CertRequestProperties"/>
        <xs:element minOccurs="0" ref="csig:SignMessage" maxOccurs="unbounded"/>
        <xs:element minOccurs="0" ref="csig:OtherRequestInfo"/>
    </xs:sequence>
    <xs:attribute name="Version" type="xs:string" use="optional" default="1.1"/>
</xs:complexType>
```

3.1.1. Type CertRequestPropertiesType

The **CertRequestPropertiesType** complex type is used to specify requested properties of the signature certificate that is associated with the generated signature.

The **CertRequestPropertiesType** complex type has the following attributes and elements:

`CertType` [Default "PKC"]

> An enumeration of certificate types, default "PKC". The supported values are "PKC", "QC" and "QC/SSCD". "QC" means that the certificate is requested to be a Qualified Certificate according to legal definitions in national law governing the issuer. "QC/SSCD" means a Qualified Certificate where the private key is declared to be residing within a Secure Signature Creation Device according to national law. "PKC" (Public Key Certificate) means a certificate that is not a Qualified Certificate.

`<saml:AuthnContextClassRef>` [Optional]

> A URI identifying the requested level of assurance that authentication of the signature certificate subject MUST comply with in order to complete signing and certificate issuance. A Signing Service MUST NOT issue signature certificates and generate the requested signature unless the authentication process used to authenticate the requested signer meets the

> requested level of assurance expressed in this element. If this element is absent, the locally configured policy of the Signing Service is assumed.

`<RequestedCertAttributes>` [Optional]

> Element holding a SAML Entity ID of an Attribute Authority that MAY be used to obtain an attribute value for the requested attribute. The EntityID value is specified using the **saml:NameIDType** complex type and MUST include a Format attribute with the value `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`.

`<OtherProperties>` [Optional]

> Other requested properties of the signature certificate.

The following schema fragment defines the **CertRequestPropertiesType** complex type:

```
<xs:complexType name="CertRequestPropertiesType">
    <xs:sequence>
        <xs:element minOccurs="0" ref="saml:AuthnContextClassRef"/>
        <xs:element minOccurs="0" ref="csig:RequestedCertAttributes"/>
        <xs:element minOccurs="0" ref="csig:OtherProperties"/>
    </xs:sequence>
    <xs:attribute default="PKC" name="CertType">
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:enumeration value="PKC"/>
                <xs:enumeration value="QC"/>
                <xs:enumeration value="QC/SSCD"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
</xs:complexType>

<xs:element name="RequestedCertAttributes"
    type="csig:RequestedAttributesType"/>
<xs:element name="OtherProperties" type="csig:AnyType"/>
```

3.1.1.1. Type RequestedAttributesType

The **RequestedAttributesType** complex type is used to represent requests for subject attributes in a signer certificate that is associated with the signer of the generated signature as a result of the sign request. This protocol is designed to accommodate the scenario where the Signing Service generates the signer key and the signer certificate at the time of signing and where the Signing Service collects the user's attributes from a SAML assertion. However, this element can also be used as a requirement for attributes in existing certificates, where the Signing Service can determine if the existing signing certificate matches the requirements of the requesting service.

The **RequestedAttributesType** has the following element:

`<RequestedCertAttribute>` [One or More]

> Information of type **MappedAttributeType** about a requested certificate attribute.

The following schema fragment defines the **RequestedAttributesType** complex type:

```
<xs:complexType name="RequestedAttributesType">
    <xs:sequence>
```

```
            <xs:element maxOccurs="unbounded" minOccurs="1"
                name="RequestedCertAttribute"
                type="csig:MappedAttributeType"/>
        </xs:sequence>
    </xs:complexType>
```

The **MappedAttributeType** complex type has the following elements and attributes:

`CertAttributeRef` [Optional]

> A reference to the certificate attribute or name type where the requester wants to store this attribute value. The information in this attribute depends on the selected `CertNameType` attribute value. If the `CertNameType` is "**rdn**" or "**sda**", then this attribute MUST contain a string representation of an object identifier (OID). If the `CertNameType` is "**san**" (Subject Alternative Name) and the target name is a GeneralName, then this attribute MUST hold a string representation of the tag value of the target GeneralName type, e.g. "1" for rfc822Name (E-mail) or "2" for dNSName. If the `CertNameType` is "**san**" and the target name form is an OtherName, then this attribute value MUST include a string representation of the object identifier of the target OtherName form.
>
> Representation of an OID as a string in this attribute MUST consist of a sequence of integers delimited by a dot. This string MUST not contain white space or line breaks. Example: "2.5.4.32".

`CertNameType` [Optional]

> An enumeration of the target name form for storing the associated SAML attribute value in the certificate. The available values are "rdn" for storing the attribute value as an attribute in a Relative Distinguished Name in the subject field of the certificate, "san" for storing the attribute value in a subject alternative name extension and "sda" for storing the attribute value in a subject directory attribute extension. The default value for this attribute is "rdn".

`FriendlyName` [Optional]

> An optional friendly name of the subject attribute, e.g. "givenName". Note that this name does not need to map to any particular naming convention and its value MUST NOT be used by the Signing Service for attribute type mapping. This name is present for display purposes only.

`DefaultValue` [Optional]

> An optional default value for the requested attribute. This value MAY be used by the Signing Service if no authoritative value for the attribute can be obtained when the Signing Service authenticates the user. This value MUST NOT be used by the Signing Service unless this value is consistent with a defined policy at the Signing Service. A typical valid use of this attribute is to hold a default countryName attribute value that matches a set of allowed countryName values. By accepting the default attribute value provided in this attribute, the Signing Service accept the requesting service as an authoritative source for this particular requested attribute.

`Required` [Optional] (Default false)

> If this attribute is set to true, the Signing Service MUST ensure that the signing certificate contains a subject attribute of the requested type, or else the Signing Service MUST NOT generate the requested signature.

`<AttributeAuthority>` [Zero or More]

> Element holding an Entity ID of an Attribute Authority that MAY be used to obtain an attribute value for the requested attribute. The EntityID value is specified using the **saml:NameIDType** complex type and MUST include a `Format` attribute with the value `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`.

`<SamlAttributeName>` [Zero or More]

> Element of type **PreferredSAMLAttributeNameType** complex type holding a name of a SAML subject attribute that is allowed to provide the content value for the requested certificate attribute.

The following schema fragment defines the **MappedAttributeType** complex type:

```
<xs:complexType name="MappedAttributeType">
    <xs:sequence>
        <xs:element maxOccurs="unbounded" minOccurs="0"
            name="AttributeAuthority" type="saml:NameIDType"/>
        <xs:element maxOccurs="unbounded" minOccurs="0"
            name="SamlAttributeName"
            type="csig:PreferredSAMLAttributeNameType"/>
    </xs:sequence>
    <xs:attribute name="CertAttributeRef" type="xs:string" use="optional"/>
    <xs:attribute name="CertNameType" default="rdn" use="optional">
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:enumeration value="rdn"/>
                <xs:enumeration value="san"/>
                <xs:enumeration value="sda"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="FriendlyName" type="xs:string"/>
    <xs:attribute name="DefaultValue" type="xs:string"/>
    <xs:attribute name="Required" type="xs:boolean" default="false"/>
</xs:complexType>
```

The **PreferredSAMLAttributeNameType** complex type holds a string value of a SAML attribute name. This attribute name SHALL be mapped against attribute names in `<saml:Attribute>` elements representing the subject in a SAML assertion that is used to authenticate the signer.

The **PreferredSAMLAttributeNameType** complex type has the following attribute:

`Order` [Optional]

> An integer specifying the order of preference of this SAML attribute. If more than one SAML attribute is listed, the SAML attribute with the lowest `Order` integer value that is present as a subject attribute in the SAML assertion, SHALL be used. SAML attributes with an absent `Order` attribute SHALL be treated as having an order value of 0. Multiple SAML attributes with an identical `Order` attribute values SHALL be treated as having equal priority.

The following schema fragment defines the **PreferredSAMLAttributeNameType** complex type:

```
<xs:complexType name="PreferredSAMLAttributeNameType">
    <xs:simpleContent>
        <xs:extension base="xs:string">
            <xs:attribute name="Order" type="xs:int" default="0"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
```

3.1.2. Element <SignMessage>

The `<SignMessage>` element holds a message to the signer with information about what is being signed. The sign message is provided either in plain text using the `<Message>` child element or as an encrypted message using the `<EncryptedMessage>` child

element. This element's **SignMessageType** complex type includes the following attributes and elements:

`MustShow` [Optional] (Default "`false`")

> When this attribute is set to `true` then the requested signature MUST NOT be created unless this message has been displayed and accepted by the signer. The default is `false`.

`DisplayEntity` [Optional]

> The EntityID of the entity responsible for displaying the sign message to the signer. When the sign message is encrypted, then this entity is also the holder of the private decryption key necessary to decrypt the sign message.

`MimeType` [Optional] (Default "`text`")

> The mime type defining the message format. This is an enumeration of the valid attribute values `text` (plain text), `text/html` (html) or `text/markdown` (markdown). This specification does not specify any particular restrictions on the provided message but it is RECOMMENDED that sign message content is restricted to a limited set of valid tags and attributes, and that the display entity performs filtering to enforce these restrictions before displaying the message. The means through which parties agree on such restrictions are outside the scope of this specification, but one valid option to communicate such restrictions could be through federation metadata.

`<Message>` [Choice]

> The base64 encoded sign message in unencrypted form. The message MUST be encoded using UTF-8.

`<EncryptedMessage>` [Choice]

> An encrypted `<Mesage>` element. Either a `<Message>` or an `<EncryptedMessage>` element MUST be present.

The following schema fragment defines the `<SignMessage>` element and the **SignMessageType** complex type:

```
<xs:complexType name="SignMessageType">
    <xs:choice>
        <xs:element ref="csig:Message"/>
        <xs:element ref="csig:EncryptedMessage"/>
    </xs:choice>
    <xs:attribute name="MustShow" type="xs:boolean" default="false"/>
    <xs:attribute name="DisplayEntity" type="xs:anyURI"/>
    <xs:attribute name="MimeType" default="text">
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:enumeration value="text/html"/>
                <xs:enumeration value="text"/>
                <xs:enumeration value="text/markdown"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>
<xs:element name="Message" type="xs:base64Binary"/>
<xs:element name="EncryptedMessage" type="saml:EncryptedElementType"/>
```

3.2. Element <SignResponseExtension>

The `<SignResponseExtension>` element is an extension point that allows a requesting service to add essential sign response information to the sign response. When present, this element MUST be included in the `<dss:OptionalOutputs>` element in a `<dss:SignResponse>` element.

This element's **SignResponseExtensionType** complex type includes the following attributes and elements:

`Version` [Optional] (Default "`1.1`")

> The version of this specification. If absent, the version value defaults to "1.0". This attribute provides means for the receiving service to determine the expected syntax of the response based on the protocol version.

`<ResponseTime>` [Required]

> The time when the sign response was created.

`<Request>` [Optional]

> A base64 encoded signed `<dss:SignRequest>` element that contains the request related to this sign response. This element MUST be present if signing was successful.

`<SignerAssertionInfo>` [Optional]

> An element of type **SignerAssertionInfoType** holding information about how the signer was authenticated by the sign service as well as information about subject attribute values present in the SAML assertion authenticating the signer, which was incorporated into the signer certificate. This element MUST be present if signing was successful.

`<SignatureCertificateChain>` [Optional]

> An element of type **CertificateChainType** holding the signer certificate as well as other certificates that may be used to validate the signature. This element MUST be present if signing was successful and MUST contain all certificates that are necessary to compile a complete and functional signed document. Certificates in `<SignatureCertificateChain>` MUST be provided in sequence with the signature certificate first followed by any CA certificates that can be used to verify the previous certificate in the sequence, ending with a self-signed root certificate.

`<OtherResponseInfo>` [Optional]

> Optional sign response elements of type **AnyType**.

The following schema fragment defines the `<SignResponseExtension>` element and the **SignResponseExtensionType** complex type:

```
<xs:element name="SignResponseExtension"
    type="csig:SignResponseExtensionType"/>
<xs:complexType name="SignResponseExtensionType">
    <xs:sequence>
        <xs:element ref="csig:ResponseTime"/>
        <xs:element minOccurs="0" ref="csig:Request"/>
        <xs:element maxOccurs="1" minOccurs="0"
            ref="csig:SignerAssertionInfo"/>
        <xs:element minOccurs="0"
            ref="csig:SignatureCertificateChain"/>
        <xs:element minOccurs="0"
            ref="csig:OtherResponseInfo"/>
    </xs:sequence>
    <xs:attribute name="Version" type="xs:string"
        default="1.1"/>
```

```
    </xs:complexType>

    <xs:element name="ResponseTime" type="xs:dateTime"/>
    <xs:element name="Request" type="xs:base64Binary"/>
    <xs:element name="SignerAssertionInfo"
        type="csig:SignerAssertionInfoType"/>
    <xs:element name="SignatureCertificateChain"
        type="csig:CertificateChainType"/>
    <xs:element name="OtherResponseInfo" type="csig:AnyType"/>
```

3.2.1. Type SignerAssertionInfoType

The **SignerAssertionInfoType** complex type has the following elements:

`<ContextInfo>` [Required]

> This element of type **ContextInfoType** holds information about SAML authentication context related to signer authentication through a SAML assertion.

`<saml:AttributeStatement>` [Required]

> This element of type **saml:AttributeStatementType** (see [*SAML2.0*]) holds subject attributes obtained from the SAML assertion used to authenticate the signer at the Signing Service. For integrity reasons, this element SHOULD only provide information about SAML attribute values that maps to subject identity information in the signer's certificate.

`<SamlAssertions>` [Zero or More]

> Any number of relevant SAML assertions that was relevant for authenticating the signer and signer's identity attributes at the Signing Service.

The following schema fragment defines the **SignerAssertionInfoType** complex type:

```
<xs:complexType name="SignerAssertionInfoType">
    <xs:sequence>
        <xs:element ref="csig:ContextInfo"/>
        <xs:element ref="saml:AttributeStatement"/>
        <xs:element minOccurs="0" ref="csig:SamlAssertions"/>
    </xs:sequence>
</xs:complexType>

    <xs:element name="ContextInfo" type="csig:ContextInfoType"/>
    <xs:element name="SamlAssertions" type="csig:SAMLAssertionsType"/>
```

3.2.1.1. Type ContextInfoType

The **ContextInfoType** complex type has the following elements:

`<IdentityProvider>`[Required]

> The EntityID of the Identity Provider that authenticated the signer to the Signing Service.

`<AuthenticationInstant>` [Required]

> The time when the Signing Service authenticated the signer.

`<saml:AuthnContextClassRef>` [Required]

> A URI reference to the authentication context class (see [*SAML2.0*]).

`<ServiceID>` [Optional]

> An arbitrary identifier of the instance of the Signing Service that authenticated the signer.

`<AuthType>` [Optional]

> An arbitrary identifier of the service used by the Signing Service to authenticate the signer (e.g. "shibboleth".)

`<AssertionRef>` [Optional]

> A reference to the assertion used to identify the signer. This MAY be the ID attribute of a `<saml:Assertion>` element but MAY also be any other reference that can be used to locate and identify the assertion.

```
<xs:complexType name="ContextInfoType">
    <xs:sequence maxOccurs="1" minOccurs="0">
        <xs:element name="IdentityProvider" type="saml:NameIDType"/>
        <xs:element name="AuthenticationInstant" type="xs:dateTime"/>
        <xs:element ref="saml:AuthnContextClassRef"/>
        <xs:element minOccurs="0" name="ServiceID" type="xs:string"/>
        <xs:element minOccurs="0" name="AuthType" type="xs:string"/>
        <xs:element minOccurs="0" name="AssertionRef" type="xs:string"/>
    </xs:sequence>
</xs:complexType>
```

3.2.1.2. Type SAMLAssertionsType

The **SAMLAssertionsType** is used to store the bytes of an arbitrary number of SAML assertions (see [*SAML2.0*]). This complex type has the following elements:

`<Assertion>`[One or More]

> One or more SAML assertions represented by the bytes of each assertion. Assertions are stored as byte data to ensure that the signature on each assertion can be validated. Assertions stored in this element MUST retain its original canonical format so that any signature on the assertion can be validated.

```
<xs:complexType name="SAMLAssertionsType">
    <xs:sequence>
        <xs:element maxOccurs="unbounded" name="Assertion"
            type="xs:base64Binary"/>
    </xs:sequence>
</xs:complexType>
```

3.2.2. Type CertificateChainType

This complex type can be used to hold a sequence of X.509 certificates. Certificates MUST be provided in sequence with the end-entity certificate first in the sequence followed by any CA certificates that can be used to verify the previous certificate in the sequence, ending with a self signed root certificate.

The **CertificateChainType** complex type has the following elements:

`<X509Certificate>` [One or More]

> An X.509 certificate [*RFC 5280*] that is part of a certificate chain that can be used to verify the generated signature. The certificate SHALL be represented as a base64Binary of the DER encoded certificate.

The following schema fragment defines the **CertificateChainType** complex type:

```
<xs:complexType name="CertificateChainType">
    <xs:sequence>
        <xs:element maxOccurs="unbounded" name="X509Certificate"
            type="xs:base64Binary"/>
    </xs:sequence>
</xs:complexType>
```

4. Extensions to <dss:InputDocuments> and <dss:SignatureObject>

This section defines elements that extends the `<dss:InputDocuments>` element of DSS sign requests and the `<dss:SignatureObject>` element of DSS sign responses by inclusion in their respective `<dss:Other>` element.

4.1. Element <SignTasks>

The `<SignTasks>` element, when present, MUST appear either in the `<dss:Other>` element of the `<dss:InputDocuments>` element of a sign request or in the `<dss:Other>` element of the `<dss:SignatureObject>` element of a sign response.

This element holds information about sign tasks that are requested in a sign request and returned in a sign response. If information about a sign task is provided using this element in a sign request, then the corresponding signature result data MUST also be provided using this element in the sign response.

This element's **SignTasksType** complex type includes the following attributes and elements:

`<SignTaskData>` [One or More]

> Input and output data associated with a sign task. A request MAY contain several instances of this element. When multiple instances of this element are present in the request, this means that the Signing Service is requested to generate multiple signatures (one for each `<SignTaskData>` element) using the same signing key and signature certificate. This allows batch signing of several different documents in the same signing instance or creation of multiple signatures on the same document such as signing XML content of a PDF document with an XML signature, while signing the rest of the document with a PDF signature.

The following schema fragment defines the `<SignTasks>` element and its **SignTasksType** complex type:

```
<xs:element name="SignTasks" type="csig:SignTasksType"/>
<xs:complexType name="SignTasksType">
    <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="csig:SignTaskData"/>
    </xs:sequence>
</xs:complexType>
<xs:element name="SignTaskData" type="csig:SignTaskDataType"/>
```

4.1.1. Element <SignTaskData>

When present in a sign request, this element provides input data to a signature generation process. When present in a sign response, this element provides the corresponding signature result data. When a request provides input data using this type of element, then an element of this type MUST also be used to return the corresponding signature result data.

This element's **SignTaskDataType** complex type includes the following attributes and elements:

`SignTaskId` [Optional]

> An identifier of the signature task that is represented by this element. If the request contains multiple instances of `<SignTaskData>` representing separate sign tasks, then each instance of the element MUST have a SignatureId attribute value that is unique among all sign tasks in the sign request. When this attribute is present, the same attribute value MUST be returned in the corresponding `<SignTaskData>` element in the response that holds corresponding signature result data.

`SigType` [Required]

> Enumerated identifier of the type of signature format the canonicalized signed information octets in the `<ToBeSignedBytes>` element are associated with. This MUST be one of the enumerated values "XML", "PDF", "CMS" of "ASiC".

`AdESType` [Optional]

> Specifies the type of AdES signature. BES means that the signing certificate hash must be covered by the signature. EPES means that the signing certificate hash and a signature policy identifier must be covered by the signature.

`ProcessingRules` [Optional]

> A URI identifying one or more processing rules that the Signing Service MUST apply when processing and using the provided signed information octets. The Signing Service MUST NOT process and complete the signature request if this attribute contains a URI that is not recognized by the Signing Service. When this attribute is present in the sign response, it represents a statement by the Signing Service that the identified processing rule was successfully executed.

`<ToBeSignedBytes>` [Required]

> The bytes to be hashed and signed when generating the requested signature. For an XML signature this MUST be the canonicalized octets of a `<dss:SignedInfo>` element. For a PDF signature this MUST be the octets of the DER encoded SignedAttrs value (signed attributes). If this data was altered by the signature process, for example as a result of changing a signing time attribute in PDF SignedAttrs, or as a result of adding a reference to a hash of the signature certificate in an XAdES signature, the altered data MUST be returned in the sign response using this element.

`<AdESObject>` [Optional]

> An element of type **AdESObjectType** complex type holding data to support generation of a signature according to any of the ETSI Advanced Electronic Signature (AdES) standard formats.

`<Base64Signature>` [Optional]

> The output signature value of the signature creation process associated with this sign task. This element's optional `Type` attribute, if present, SHALL contain a URI indicating the signature algorithm that was used to generate the signature value.

`<OtherSignTaskData>` [Optional]

> Other input or output data elements associated with the sign task.

The following schema fragment defines the `<SignTaskData>` element and its **SignTaskDataType** complex type:

```
<xs:element name="SignTaskData" type="csig:SignTaskDataType"/>
<xs:complexType name="SignTaskDataType">
    <xs:sequence>
        <xs:element ref="csig:ToBeSignedBytes"/>
        <xs:element maxOccurs="1" minOccurs="0" ref="csig:AdESObject"/>
        <xs:element minOccurs="0" ref="csig:Base64Signature"/>
        <xs:element minOccurs="0" ref="csig:OtherSignTaskData"/>
    </xs:sequence>
    <xs:attribute name="SignTaskId" type="xs:string"/>
    <xs:attribute name="SigType" use="required">
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:enumeration value="XML"/>
                <xs:enumeration value="PDF"/>
                <xs:enumeration value="CMS"/>
```

```
                <xs:enumeration value="ASiC"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
    <xs:attribute default="None" name="AdESType">
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:enumeration value="None"/>
                <xs:enumeration value="BES"/>
                <xs:enumeration value="EPES"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="ProcessingRules" type="xs:anyURI" use="optional"/>
</xs:complexType>

<xs:element name="ToBeSignedBytes" type="xs:base64Binary"/>
<xs:element name="AdESObject" type="csig:AdESObjectType"/>
<xs:element name="Base64Signature" type="csig:Base64SignatureType"/>
<xs:element name="OtherSignTaskData" type="csig:AnyType"/>
```

4.1.1.1. Type AdESObjectType

The **AdESObjectType** complex type holds a base64 encoded object that is referenced from the signature and which content has to be modified as part of the certificate and signature generation process. Handling of this data as in the signature generation process MAY be affected by rules identifyed by the `ProcessingRules`, `SigType` and `AdESType` attributes of the parent `<SignatureInputObjects>` elements.

This complex type has the following elements:

`<SignatureId>` [Optional]

> An optional identifier of the signature. When the requested signature is a XAdES signature, this is the value of the Id attribute of the `<ds:Signature>` element of the resulting signature, which is used to construct the `Target` attribute of the `<xades:QualifyingProperties>` element.

`<AdESObjectBytes>` [Optional]

> A base64 encoded object that is referenced from the signature and which content has to be modified as part of the certificate and signature generation process. The type of data in the base64 encoded bytes and the rules of handling of this data as in the signature generation process is determined through the `ProcessingRules`, `SigType` and `AdESType` attributes of the parent `<SignatureInputObjects>` elements. When the signature type is XAdES, then this is the bytes of the `<ds:Object>` element that holds the `<xades:QualifyingProperties>` element of the signature where the hash of the signature certificate will be placed.
>
> When this element is present in the request, it forms a base for construction of this object in the signature process.

`<OtherAdESData>` [Optional]

> Other input data related to the AdES signature generation process.

The following schema fragment defines the **AdESObjectType** complex type

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

```
            <xs:element minOccurs="0" name="AdESObjectBytes"
                type="xs:base64Binary"/>
            <xs:element minOccurs="0" name="OtherAdESData" type="csig:AnyType"/>
        </xs:sequence>
    </xs:complexType>
```

5. Signing sign requests and responses

This specification supports a scenario where a requesting service requests the signer to sign some data and where the same requesting service receives the signature from the Signing Service. In this scenario the requesting service acts as the presenter of information to be signed to the signer. Implementers of this specification MUST sign requests and responses for signature creation to protect against spoofing and substitution attacks. If a hash of the document to be signed is replaced in a sign request, the signer may end up signing something completely different than what the requesting service presented to the signer.

When a `<dss:SignRequest>` is signed, the signature of that request MUST be placed as the last child element in the `<dss:OptionalInputs>` element. The Signing Service MUST check this signature and MUST check that the signature covers all data in the `<dss:SignRequest>` element (except for the signature itself).

When a `<dss:SignResponse>` is signed, the signature of that response MUST be placed as the last child element in the `<dss:OptionalOutputs>` element. The Signing Service MUST check this signature and MUST check that the signature covers all data in the `<dss:SignResponse>` element (except for the signature itself)

Appendix A. XML Schema

This section provides the full XML Schema declaration for the DSS protocol extension defined in this document. In case of differences between the XML Schema in this appendix and XML Scema fragments in the sections above, the XML Schema in this appendix is the normative one.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
    targetNamespace="http://id.elegnamnden.se/csig/1.1/dss-ext/ns"
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:dss="urn:oasis:names:tc:dss:1.0:core:schema"
    xmlns:csig="http://id.elegnamnden.se/csig/1.1/dss-ext/ns">
    <xs:import namespace="urn:oasis:names:tc:SAML:2.0:assertion"
        schemaLocation="saml-schema-assertion-2.0.xsd"/>
    <xs:element name="SignRequestExtension" type="csig:SignRequestExtensionType">
        <xs:annotation>
            <xs:documentation>Extension to an OASIS DSS SignRequest, providing additional
information about a sign request. This element extends the
dss:OptionalInputs element of a dss:SignRequest.</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="SignResponseExtension" type="csig:SignResponseExtensionType">
        <xs:annotation>
            <xs:documentation>Extension to an OASIS DSS SignResponse, providing additional information
about a sign response. This element extends the dss:OptionalOutput element
of a dss:SignResponse.</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="SignTasks" type="csig:SignTasksType"/>
    <xs:element name="SignTaskData" type="csig:SignTaskDataType"/>
    <xs:element name="RequestTime" type="xs:dateTime">
        <xs:annotation>
            <xs:documentation>Time when the request was created</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="Signer" type="saml:AttributeStatementType">
        <xs:annotation>
            <xs:documentation>The identity of the signer expressed as a sequence of SAML attributes
using the AttributesType complex type.</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="IdentityProvider" type="saml:NameIDType">
        <xs:annotation>
            <xs:documentation>The SAML EntityID of the Identity Provider that MUST be used to
authenticate the signer before signing. The EntitID value is specified
using the saml:NameIDType complex type and MUST include a Format
attribute with the value urn:oasis:names:tc:SAML:2.0:nameid-format:entity.</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="SignRequester" type="saml:NameIDType">
        <xs:annotation>
            <xs:documentation>The SAML EntityID of the service that sends this request to the signing service.
The EntityID value is specified using the saml:NameIDType complex type and MUST
include a Format attribute with the value
urn:oasis:names:tc:SAML:2.0:nameid-format:entity.</xs:documentation>
        </xs:annotation>
    </xs:element>
```

```
    <xs:element name="SignService" type="saml:NameIDType">
        <xs:annotation>
            <xs:documentation>The SAML EntityID of the service to which this Sign Request is sent.
The EntityID value is specified using the saml:NameIDType complex type
and MUST include a Format attribute with the value
urn:oasis:names:tc:SAML:2.0:nameid-format:entity.</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="RequestedSignatureAlgorithm" type="xs:anyURI">
        <xs:annotation>
            <xs:documentation>An identifier of the signature algorithm the requesting service prefers
when generating the requested signature.</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="CertRequestProperties" type="csig:CertRequestPropertiesType">
        <xs:annotation>
            <xs:documentation>The requested properties of the signature certificate being issued by the
signature service.</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="RequestedCertAttributes" type="csig:RequestedAttributesType">
        <xs:annotation>
            <xs:documentation>An optional set of requested attributes that the requesting service prefers
or requires in the subject name of the generated signing certificate.</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="OtherProperties" type="csig:AnyType"/>
    <xs:element name="SignMessage" type="csig:SignMessageType">
        <xs:annotation>
            <xs:documentation>Sign message included as a choice of a Base64 encoded string or an ecrypted sign message.</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="Message" type="xs:base64Binary"/>
    <xs:element name="EncryptedMessage" type="saml:EncryptedElementType"/>
    <xs:element name="OtherRequestInfo" type="csig:AnyType">
        <xs:annotation>
            <xs:documentation>Any additional inputs to the request extension.</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="ResponseTime" type="xs:dateTime">
        <xs:annotation>
            <xs:documentation>The time when the sign response was created.</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="Request" type="xs:base64Binary">
        <xs:annotation>
            <xs:documentation>An element of type EncodedRequestType with base64Binary base type, holding
a representation of a complete and signed dss:SignRequest element that is
related to this sign response. This element MUST be present if signing was
successful.</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="SignerAssertionInfo" type="csig:SignerAssertionInfoType">
        <xs:annotation>
            <xs:documentation>An element of type SignerAssertionInfoType holding information about how
the signer was authenticated by the sign service as well as information
about subject attribute values present in the SAML assertion authenticating
the signer, which was incorporated into the signer certificate. This element
MUST be present if signing was successful.</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="ContextInfo" type="csig:ContextInfoType"/>
    <xs:element name="SamlAssertions" type="csig:SAMLAssertionsType"/>
```

```
    <xs:element name="SignatureCertificateChain" type="csig:CertificateChainType">
        <xs:annotation>
            <xs:documentation>An element of type CertificateChainType holding the signer certificate as
well as other certificates that may be used to validate the signature. This
element MUST be present if signing was successful and MUST contain all
certificate that are necessary to compile a complete and functional signed
document.</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="OtherResponseInfo" type="csig:AnyType">
        <xs:annotation>
            <xs:documentation>Optional sign response elements of type AnyType.</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="ToBeSignedBytes" type="xs:base64Binary">
        <xs:annotation>
            <xs:documentation>The octets that are hashed and signed when generating the signture. For
PDF and common modes of CMS this is the DER encoded SignedAttributess field.
For XML this is the canonicalized SignedInfo octets.</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="AdESObject" type="csig:AdESObjectType">
        <xs:annotation>
            <xs:documentation>Information in support of AdES signature creation</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="Base64Signature" type="csig:Base64SignatureType">
        <xs:annotation>
            <xs:documentation>Result signature bytes</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="OtherSignTaskData" type="csig:AnyType"/>
    <xs:complexType name="SignRequestExtensionType">
        <xs:sequence>
            <xs:element ref="csig:RequestTime"/>
            <xs:element ref="saml:Conditions">
                <xs:annotation>
                    <xs:documentation>Conditions that MUST be evaluated when assessing the validity of and/or
when using the Sign Request. See Section 2.5 of [SAML2.0]for additional
information on how to evaluate conditions.

This element MUST include the attributes NotBefore and NotOnOrAfter and
MUST include the element saml:AudienceRestriction which in turn MUST
contain one saml:Audience element, specifying the return URL for any
resulting Sign Response message.</xs:documentation>
                </xs:annotation>
            </xs:element>
            <xs:element ref="csig:Signer" minOccurs="0"/>
            <xs:element ref="csig:IdentityProvider"/>
            <xs:element ref="csig:SignRequester"/>
            <xs:element ref="csig:SignService"/>
            <xs:element minOccurs="0" ref="csig:RequestedSignatureAlgorithm"/>
            <xs:element minOccurs="0" ref="csig:CertRequestProperties"/>
            <xs:element minOccurs="0" ref="csig:SignMessage" maxOccurs="1"/>
            <xs:element minOccurs="0" ref="csig:OtherRequestInfo"/>
        </xs:sequence>
        <xs:attribute name="Version" type="xs:string" use="optional" default="1.1">
            <xs:annotation>
                <xs:documentation>The version of this specification. If absent, the version value defaults to "1.0".
This attribute provide means for the receiving service to determine the
expected syntax of the response based on protocol version.</xs:documentation>
            </xs:annotation>
        </xs:attribute>
```

```
    </xs:complexType>
    <xs:complexType name="SignResponseExtensionType">
        <xs:sequence>
            <xs:element ref="csig:ResponseTime"/>
            <xs:element minOccurs="0" ref="csig:Request"/>
            <xs:element maxOccurs="1" minOccurs="0" ref="csig:SignerAssertionInfo"/>
            <xs:element minOccurs="0" ref="csig:SignatureCertificateChain"/>
            <xs:element minOccurs="0" ref="csig:OtherResponseInfo"/>
        </xs:sequence>
        <xs:attribute name="Version" type="xs:string" default="1.1">
            <xs:annotation>
                <xs:documentation>The version of this specification. If absent, the version value defaults to "1.0".
This attribute provide means for the receiving service to determine the
expected syntax of the response based on protocol version.</xs:documentation>
            </xs:annotation>
        </xs:attribute>
    </xs:complexType>
    <xs:complexType name="CertificateChainType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" name="X509Certificate" type="xs:base64Binary"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="MappedAttributeType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0" name="AttributeAuthority"
                type="saml:NameIDType"/>
            <xs:element maxOccurs="unbounded" minOccurs="0" name="SamlAttributeName"
                type="csig:PreferredSAMLAttributeNameType"/>
        </xs:sequence>
        <xs:attribute name="CertAttributeRef" type="xs:string" use="optional"/>
        <xs:attribute name="CertNameType" default="rdn" use="optional">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="rdn"/>
                    <xs:enumeration value="san"/>
                    <xs:enumeration value="sda"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="FriendlyName" type="xs:string"/>
        <xs:attribute name="DefaultValue" type="xs:string"/>
        <xs:attribute name="Required" type="xs:boolean" default="false"/>
    </xs:complexType>
    <xs:complexType name="RequestedAttributesType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="1" name="RequestedCertAttribute"
                type="csig:MappedAttributeType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="AnyType">
        <xs:sequence>
            <xs:any processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="SAMLAssertionsType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" name="Assertion" type="xs:base64Binary"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="PreferredSAMLAttributeNameType">
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="Order" type="xs:int" default="0"/>
```

```
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <xs:complexType name="SignTasksType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" ref="csig:SignTaskData"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="SignTaskDataType">
        <xs:sequence>
            <xs:element ref="csig:ToBeSignedBytes"/>
            <xs:element maxOccurs="1" minOccurs="0" ref="csig:AdESObject"/>
            <xs:element minOccurs="0" ref="csig:Base64Signature"/>
            <xs:element minOccurs="0" ref="csig:OtherSignTaskData"/>
        </xs:sequence>
        <xs:attribute name="SignTaskId" type="xs:string">
            <xs:annotation>
                <xs:documentation>A distinguishing id of this sign task which is used to distinguish between
multiple sign tasks in the same request</xs:documentation>
            </xs:annotation>
        </xs:attribute>
        <xs:attribute name="SigType" use="required">
            <xs:annotation>
                <xs:documentation>Enumeration of the type of signature the canonical signed information is
associated with.</xs:documentation>
            </xs:annotation>
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="XML"/>
                    <xs:enumeration value="PDF"/>
                    <xs:enumeration value="CMS"/>
                    <xs:enumeration value="ASiC"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute default="None" name="AdESType">
            <xs:annotation>
                <xs:documentation>Specifies the type of AdES signature. BES means that the signing certificate
hash must be covered by the signature. EPES means that the signing
certificate hash and a signature policy identifier must be covered by
the signature.</xs:documentation>
            </xs:annotation>
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="None"/>
                    <xs:enumeration value="BES"/>
                    <xs:enumeration value="EPES"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="ProcessingRules" type="xs:anyURI" use="optional">
            <xs:annotation>
                <xs:documentation>An identifier for processing rules that must be executed by the signature
service when processing data in this element.</xs:documentation>
            </xs:annotation>
        </xs:attribute>
    </xs:complexType>
    <xs:complexType name="AdESObjectType">
        <xs:sequence>
            <xs:element minOccurs="0" name="SignatureId" type="xs:string"/>
            <xs:element minOccurs="0" name="AdESObjectBytes" type="xs:base64Binary"/>
            <xs:element minOccurs="0" name="OtherAdESData" type="csig:AnyType"/>
        </xs:sequence>
```

```
    </xs:complexType>
    <xs:complexType name="CertRequestPropertiesType">
        <xs:sequence>
            <xs:element minOccurs="0" ref="saml:AuthnContextClassRef">
                <xs:annotation>
                    <xs:documentation>The a URI reference to the requested level of assurance with which the
certificate subject should be authenticated.</xs:documentation>
                </xs:annotation>
            </xs:element>
            <xs:element minOccurs="0" ref="csig:RequestedCertAttributes"/>
            <xs:element minOccurs="0" ref="csig:OtherProperties"/>
        </xs:sequence>
        <xs:attribute default="PKC" name="CertType">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="PKC"/>
                    <xs:enumeration value="QC"/>
                    <xs:enumeration value="QC/SSCD"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
    </xs:complexType>
    <xs:complexType name="SignerAssertionInfoType">
        <xs:sequence>
            <xs:element ref="csig:ContextInfo"/>
            <xs:element ref="saml:AttributeStatement"/>
            <xs:element minOccurs="0" ref="csig:SamlAssertions"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="ContextInfoType">
        <xs:sequence maxOccurs="1" minOccurs="0">
            <xs:element name="IdentityProvider" type="saml:NameIDType"/>
            <xs:element name="AuthenticationInstant" type="xs:dateTime"/>
            <xs:element ref="saml:AuthnContextClassRef"/>
            <xs:element minOccurs="0" name="ServiceID" type="xs:string"/>
            <xs:element minOccurs="0" name="AuthType" type="xs:string"/>
            <xs:element minOccurs="0" name="AssertionRef" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="Base64SignatureType">
        <xs:simpleContent>
            <xs:extension base="xs:base64Binary">
                <xs:attribute name="Type" type="xs:anyURI"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <xs:complexType name="SignMessageType">
        <xs:choice>
            <xs:element ref="csig:Message"/>
            <xs:element ref="csig:EncryptedMessage"/>
        </xs:choice>
        <xs:attribute name="MustShow" type="xs:boolean" default="false"/>
        <xs:attribute name="DisplayEntity" type="xs:anyURI"/>
        <xs:attribute name="MimeType" default="text">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="text/html"/>
                    <xs:enumeration value="text"/>
                    <xs:enumeration value="text/markdown"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
```

```
    </xs:complexType>
</xs:schema>
```