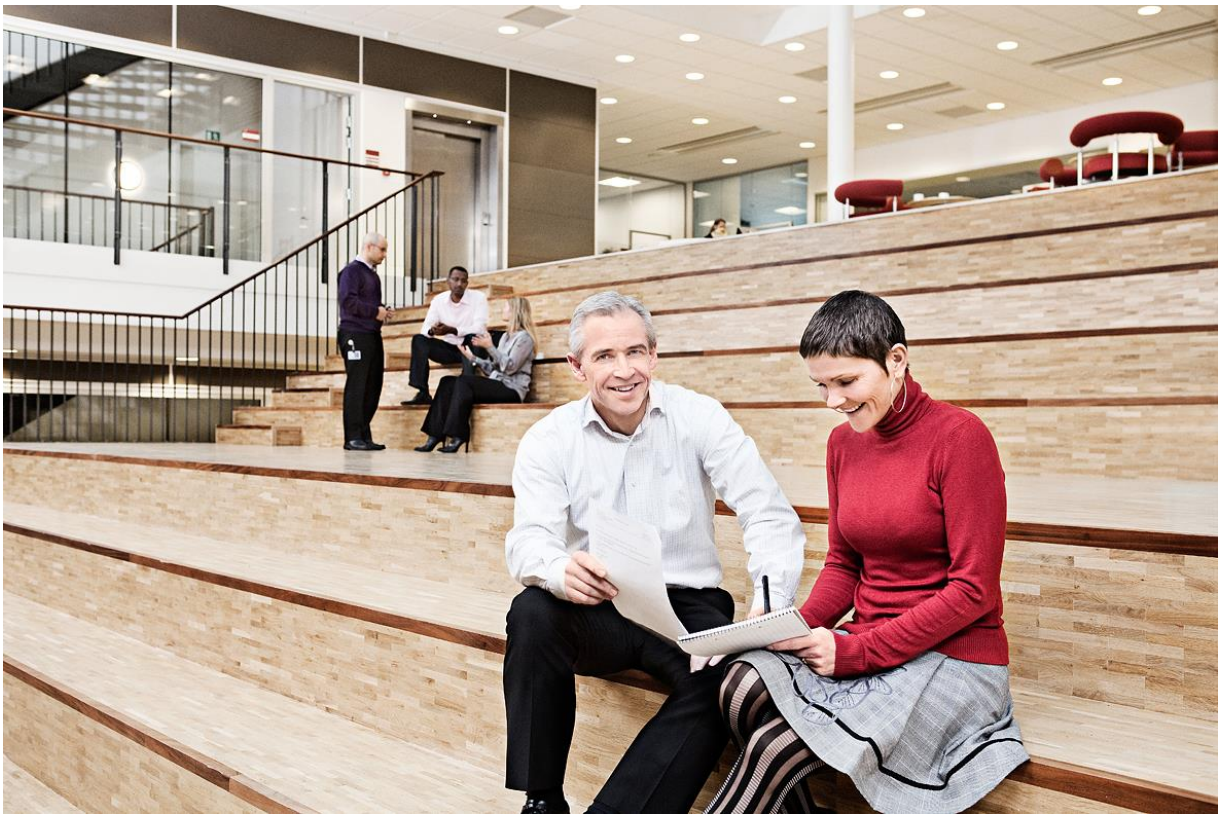


Digitaliseringsstyrelsen

KFOBS
Detail specification

4.4.5.4 Security Token Service DS – Processing rules

Version: 2.0

ID: 32309

2014-04-04

NNIT
Conscience driven. Value adding

Table of Contents

1 Introduction

This document mandate requirements to message interface, processing and formatting that Nemlog-in STS must comply with. The document operationalize requirements that originate from the Danish WS-Trust 1.3 [WST] interoperability profile [OIO-WST] and the associated deployment profile [OIO-WST-DEPL]. Some requirements are specific to Nemlog-in STS and mandated by the KFOBS requirements specification [BILAG3].

Other parties affected by this document are:

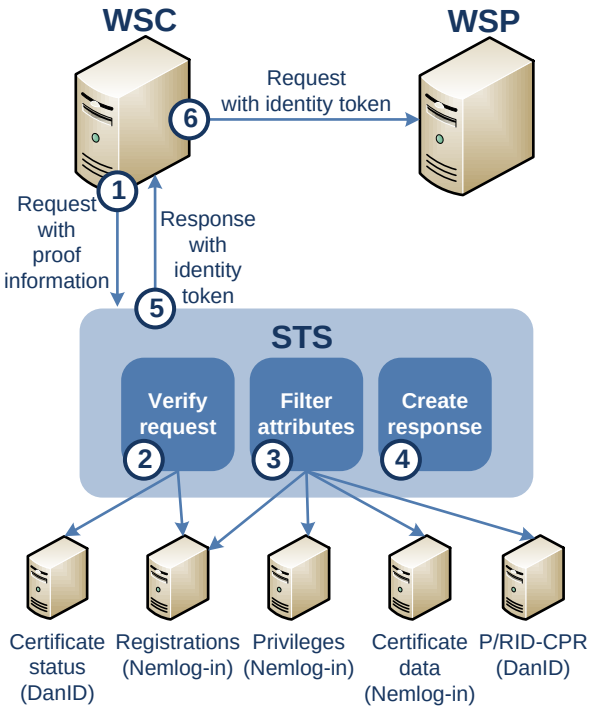
- Web service consumers (WSC) that must comply with requirements covering request and response message interfaces.
- Web service providers (WSP) that must comply with requirements covering the identity token returned in response messages (as identity tokens are subsequently delivered to WSP's using the Liberty Basic SOAP binding).

2 Processing rules

2.1 Basic processing model

WSC's exchange security tokens at STS that again can be presented for WSP's to access specific web services.

The figure below illustrates the basic processing model:



Request proof information must identify the user a request is made on behalf-of and the authenticity of the request. How proof information is established and verified depends on the usage scenario, which is described in section 2.2 "Usage scenarios".

The basic processing steps, which covers all usage scenarios:

1) WSC sends a request to STS that includes proof information.
2) STS receives and validates the request:
    a) The request message formatting is verified.
    b) The proof information is verified.
    c) The WSP the identity token is requested for must be trusted, which is verified by STS.
3) STS filters attribute values to be embedded with the identity token:

a) The WSP the identity token is requested for mandate the attributes it will accept according to WSP registrations.

4) STS issues a response message and embeds the identity token.

5) STS sends a response to WSC that includes the identity token.

6) WSC sends a request to WSP that includes the identity token.

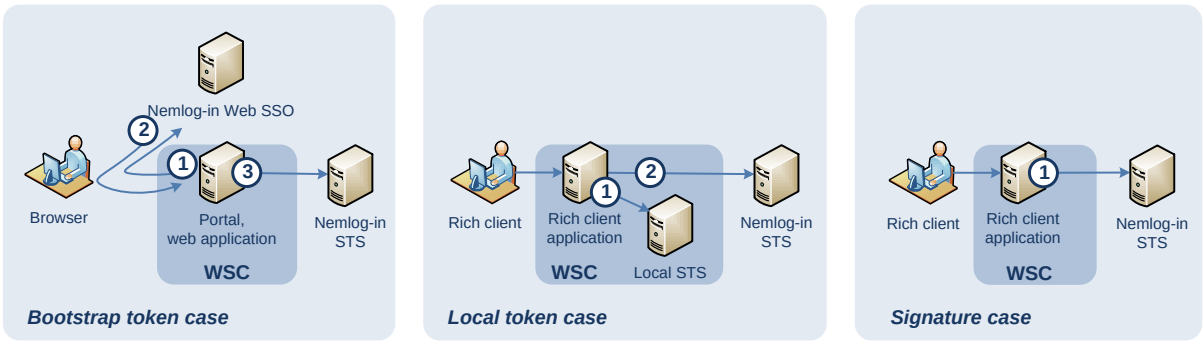
How WSC's and WSP's verify identity tokens and how these parties interact is out-of-scope from this specification.

The sections 2.3 "Request message", 2.4 "Attribute filtering" and 2.5 "Response message" details the basic processing model requirements.

2.2 Usage scenarios

In accordance with the KFOBS requirements specification [BILAG3], STS must support three usage scenarios. Each scenario instructs how WSC's obtains proof information to be presented for STS and consequently how STS must process the request prior to token exchange.

The figure below illustrates the scenarios:



The WSC and WSP parties involved must establish trust relationship with STS prior to token exchange. Trust is established through registration in Nemlog-in CSS (Connection Support System) and the type of trust required depends on the usage scenario. The registration process will be described in separate documentation.

2.2.1 Bootstrap token case

In this scenario the user identity is proofed by a bootstrap token that is obtained from Nemlog-in Web SSO as a result of a completed login transaction. The bootstrap token is embedded with the SAML assertion that is returned from Nemlog-in to WSC, who in this context assumes the role of Service Provider (SP), in accordance with the OIO SAML Web SSO profile [OIO-WEB-SSO].

Scenario authenticity:

- The bootstrap token is signed by Nemlog-in (2) and the certificate used for signing the AuthnRequest (1) to Nemlog-in Web SSO is referenced as "holder-of-key".
- The WSC must sign the request to STS (3) with the same certificate used for signing the AuthnRequest (1) to Nemlog-in Web SSO.

2.2.2 Local token case

In this scenario the user identity is proofed by a bootstrap token that is obtained from WSC's local STS hosted by the WSC organization itself. The local STS must be trusted[1] by Nemlog-in STS. The circumstances on how bootstrap tokens are issued are left to local policy, but it's assumed that users have authenticated against the organization network using e.g. Kerberos or similar protocol.

Depending on local STS policy, WSC can choose between using "bearer" or "holder-of-key" methods when presenting requests for Nemlog-in STS.

"Bearer" confirmation method:

- The bootstrap token is signed (1) by the local STS using a VOCES or FOCES certificate that is trusted by Nemlog-in STS. The token is referenced as "bearer".
- The WSC must sign the request to STS (2) using any VOCES or FOCES certificate that is bound to the same CVR-number as the local STS signing certificate.

"Holder-of-key" confirmation method:

- The bootstrap token is signed (1) by the local STS using a VOCES or FOCES certificate that is trusted by Nemlog-in STS. The certificate that is expected to be used for signing the request is referenced as "holder-of-key.
- The WSC must sign the request to STS (2) using the certificate referenced as "holder-of-key", which must be a VOCES or FOCES certificate that is bound to the same CVR-number as the local STS signing certificate.

2.2.3 Signature case

In this scenario the user identity is proofed by the user signing the request to Nemlog-in STS. The scenario contains no bootstrap tokens. The WSC constitutes in this scenario any application hosted by any organization and no trust is established directly between WSC and Nemlog-in STS.

Scenario authenticity:

- The user must sign (1) the request to STS using his/her MOCES certificate that is bound to a CVR-number trusted by STS[2].

[1] The local STS must be registered as WSC in Nemlog-in CSS and enter terms and conditions with Digitaliseringsstyrelsen.
[2] CVR-number must be registered as user organization in Nemlog-in CSS.

2.3 Request message

This section specifies requirements related to request message interface, processing and formatting, which STS and subsequent WSC's must adhere to.

General request message requirements

A request must comply with the following general requirements:

- STS supports the WS-Trust issuance binding only.
- STS supports requests for one identity token per request only.
- STS requires that requests identify the user the request is made on behalf of in accordance with 2.2 "Usage scenarios".
- STS requires that SOAP request and embedded bootstrap token, if any, are signed in accordance with 2.2 "Usage scenarios".
- STS supports user identities with requests only (system identities are not supported at this time).
- STS issues identity tokens for trusted[3] WSP's only.
- STS does not support and will ignore not explicitly requested attributes (`wst:Claims`) in requests.

Request message requirements

For bootstrap token case and local token case scenarios, the request message must contain the elements shown below:

```
<S11:Envelope>
  <S11:Header>
    <wsa:Action>...</wsa:Action>
    <wsa:MessageID>...</wsa:MessageID>
    <wsa:To>...</wsa:To>
    <wsse:Security>...</wsse:Security>
  </S11:Header>
  <S11:Body>
    <wst:RequestSecurityToken>
       <wst:RequestType>...</wst:RequestType>
       <wst14:ActAs>
         <saml2:Assertion>...</saml2:Assertion>
       </wst14:ActAs>
       <wsp:AppliesTo>...</wsp:AppliesTo>
    </wst:RequestSecurityToken>
  </S11:Body>
</S11:Envelope>
```

[3] WSP's must be registered in Nemlog-in CSS, which includes technical data about the service and the attributes requested by the service.

For signature case scenario, the request message must contain the elements shown below:

```
<S11:Envelope>
  <S11:Header>
    <wsa:Action>...</wsa:Action>
    <wsa:MessageID>...</wsa:MessageID>
    <wsa:To>...</wsa:To>
    <wsse:Security>...</wsse:Security>
  </S11:Header>
  <S11:Body>
    <wst:RequestSecurityToken>
       <wst:RequestType>...</wst:RequestType>
       <wsp:AppliesTo>...</wsp:AppliesTo>
    </wst:RequestSecurityToken>
  </S11:Body>
</S11:Envelope>
```

2.3.1 SOAP Header

This section specifies requirements related to the SOAP request header block.

General processing rules:

- SOAP header may contain any elements and attributes defined by the respective schemas used in headers, but STS is only obligated to obey with the conditions explicitly specified in this section. Thus, elements and attributes not covered by conditions are normally ignored (except when verifying signature).
- SOAP header must exclusively obey to the conditions explicitly specified in this section otherwise the request will be rejected.

2.3.1.1 /wsa:Action

The element is required.

The `Action` element must contain an URI identifier that uniquely (and opaquely) identifies the semantics implied by this message.

STS uses the issuance binding and the element must therefore be set to the following URI: http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/Issue

The `Action` element must conform to the normative example below:

```
<wsa:Action>
  http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/Issue
</wsa:Action>
```

2.3.1.2 /wsa:MessageID

The element is required.

The `MessageID` element uniquely and opaquely identifies the SOAP request and is used for correlating the SOAP response to this. The value of `MessageID` must be an opaque URI that must comply with the following convention: ”uuid:”+*unique identifier*

`MessageID` element formatting example:

```
<wsa:MessageID>
  uuid:aaaabbbb-cccc-dddd-eeee-ffffffffffff
</wsa:MessageID>
```

WS-Adressing [WSA] does not define any requirements for creating the unique identifier and its recommended that the unique identifier follows the [xs:ID] simple type.

General `MessageID` processing rules:

- WSC is responsible for creating *unique* MessageID identifiers (STS does not verify uniqueness).
- The `MessageID` identifier must not be identical with any other identifiers created for the SOAP header or body block.

2.3.1.3 /wsa:To

The element is required.

The `To` element refers the destination of the SOAP request receiver and must therefore refer the entityId of STS.

STS accepts one of three entityId’s depending on the request usage scenario:

- Bootstrap token case: https://bootstrap.sts.nemlog-in.dk
- Local token case: https://local.sts.nemlog-in.dk
- Signature case: https://signature.sts.nemlog-in.dk

`To` element formatting example:

```
<wsa:To>
    https://bootstrap.sts.nemlog-in.dk
</wsa:To>
```

2.3.1.4 /wsse:Security

The element is required.

The `Security` element must contain the SOAP request signature and the certificate used to sign the SOAP request.

The `Security` element must conform to the normative example below:

```
<wsse:Security mustUnderstand="1">
```

```
   <wsu:Timestamp>...</wsu:Timestamp>
   <wsse:BinarySecurityToken >...</wsse:BinarySecurityToken>
   <ds:Signature>...</ds:Signature>
</wsse:Security>
```

The `Security` element must contain the `mustUnderstand` attribute set to true.

2.3.1.4.1 /wsse:Security/wsu:Timestamp

The element is required.

The `Timestamp` element provides a mechanism for expressing the creation and expiration times of the signature in the SOAP request.

The `Timestamp` element may contain one `wsu:Created` element and must contain one `wsu:Expires` element. All time instants must be in UTC format as specified by the XML Schema type [xs:dateTime] (YYYY-MM-DDThh:mm:ssZ).

`Timestamp` element formatting and normative example:

```
<wsu:Timestamp wsu:Id="...">
  <wsu:Created>2004-05-06T22:04:34Z</wsu:Created>
  <wsu:Expires>2004-05-06T22:04:34Z</wsu:Expires>
</wsu:Timestamp>
```

General processing rules:

- `Expires` must be later than `Created` and after STS current time.
- `Created` should not differ substantially from the SOAP request transmission time, which must be governed by STS.
- If time instant rules are not met the SOAP request is rejected.

2.3.1.4.2 /wsse:Security/wsse:BinarySecurityToken

The element is required.

The `BinarySecurityToken` element holds the security token used for signing the SOAP request. The security token must be a VOCES, FOCES or MOCES certificate depending on the usage scenario.

Certificate requirements are specified in the "General signature processing rules" and "Usage scenario processing rules" in section 2.3.1.4.3 "/wsse:Security/wsse:Signature".

The `BinarySecurityToken` element must conform to the normative example below:

```
<wsse:BinarySecurityToken ValueType="..." EncodingType="...">
</wsse:BinarySecurityToken>
```

The `BinarySecurityToken` element must contain a `ValueType` attribute set to: http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3.

The `BinarySecurityToken` element may contain an `EncodingType` attribute set to: http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary (this is also the default value, if not explicitly stated).

The `BinarySecurityToken` element must contain the base64 encoded certificate public key used for signing the SOAP request.

`BinarySecurityToken` element formatting and normative example:

```
<wsse:BinarySecurityToken
   ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"
    EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary">
        <!-- X509 signing certificate -->
</wsse:BinarySecurityToken>
```

2.3.1.4.3 /wsse:Security/ds:Signature

The element is required.

The `Signature` element must contain the SOAP request signature. The signature must cover all SOAP headers (excluding the `ds:Signature` element itself but including header elements STS does not recognize or otherwise would ignore) and the complete SOAP body.

The `Signature` element must conform to the normative example below:

```
<ds:Signature>
  <ds:SignedInfo>
    <ds:CanonicalizationMethod Algorithm="..."/>
    <ds:SignatureMethod Algorithm="..."/>
    <!-- references to elements included in signature -->
    <ds:Reference URI="#action">...</ds:Reference>
    <ds:Reference URI="#messageid">...</ds:Reference>
    <ds:Reference URI="#to">...</ds:Reference>
    <ds:Reference URI="#replyto">...</ds:Reference>
    <ds:Reference URI="#timestamp">...</ds:Reference>
    <ds:Reference URI="#binarysecuritytoken">...</ds:Reference>
    <ds:Reference URI="#body">...</ds:Reference>
  </ds:SignedInfo>
  <ds:SignatureValue>...</ds:SignatureValue>
  <ds:KeyInfo>
     <wsse:SecurityTokenReference>
        <wsse:Reference URI="#SigningToken"/>
     </wsse:SecurityTokenReference>
  </ds:KeyInfo>
</ds:Signature>
```

SOAP requests must explicitly include the individual elements to be signed using the `Reference` element. All referenced elements must therefore carry a `wsu:Id` attribute. The `KeyInfo` element must contain a `SecurityTokenReference` element that references the signing certificate in `BinarySecurityToken`.

Each `Reference` element must conform to the normative example below:

```
<ds:Reference URI="...">
  <ds:Transforms>
    <ds:Transform Algorithm="..."/>
  </ds:Transforms>
  <ds:DigestMethod Algorithm="..."/>
  <ds:DigestValue>
     <!-- digest value -->
  </ds:DigestValue>
</ds:Reference>
```

The signature builds on XML Signature [XMLSIG] and must therefore obey with the same algorithm requirements as those specified in the XML Signature specification.

The following signature algorithms are supported by STS:

Signature element	Algorithm
`CanonicalizationMethod`	http://www.w3.org/2001/10/xml-exc-c14n#
`SignatureMethod`	http://www.w3.org/2000/09/xmldsig#rsa-sha1 http://www.w3.org/2001/04/xmldsig-more#rsa-sha256
`Transforms`	http://www.w3.org/2001/10/xml-exc-c14n#
`DigestMethod`	http://www.w3.org/2000/09/xmldsig#sha1 http://www.w3.org/2001/04/xmlenc#sha256

General signature processing rules:

The SOAP request signing certificate and signature must obey with the general rules below regardless of usage scenario:

- The signing certificate must be a VOCES, FOCES or MOCES certificate depending on the usage scenario (see "Usage scenario processing rules" below).
- The signing certificate validity period must be valid.
- The signing certificate must not be revoked.
- The signing certificate must be issued directly by a DanID CA certificate (any certificates between the root certificate and the signing certificate must be DanID CA certificates).
- The DanID CA certificate(s) must not be revoked.
- The signature must verify with the certificate contained in /wsse:Security/wsse:`BinarySecurityToken` element.

- The signing certificate must obey with rules defined in "Usage scenario processing rules" below.

Usage scenario processing rules:

In addition to the general signature processing rules, the SOAP request signing certificate must obey with rules below depending on usage scenario:

Usage scenario	Processing rule
Bootstrap token case	The SOAP request signing certificate: • Must be a VOCES or FOCES certificate. • Must correspond to the "holder-of-key" certificate contained in the `SubjectConfirmationData` element in the embedded bootstrap token.
Local token case ("holder-of-key")	The SOAP request signing certificate: • Must be a VOCES or FOCES certificate. • Must correspond to the "holder-of-key" certificate contained in the `SubjectConfirmationData` element in the embedded bootstrap token. • Must be bound to the same CVR-number as the certificate used for signing the embedded bootstrap token.
Local token case ("bearer")	The SOAP request signing certificate: • Must be a VOCES or FOCES certificate. • Must be bound to the same CVR-number as the certificate used for signing the embedded bootstrap token.
Signature case	The SOAP request signing certificate: • Must be a MOCES certificate. • Must be bound to a CVR-number that is registered as User Organization in Nemlog-in CSS.

2.3.2 SOAP Body

This section specifies requirements related to the SOAP request body block.

General processing rules:

- SOAP body must contain exactly one `<wst:RequestSecurityToken>` element.
- SOAP body may contain any elements and attributes defined by the respective schemas used in `<wst:RequestSecurityToken>`, but STS is only obligated to obey

with the conditions explicitly specified in this section. Thus, elements and attributes not covered by conditions are normally ignored (except when verifying signature).

- SOAP body must exclusively obey to the conditions explicitly specified in this section otherwise the request will be rejected.

The following abbreviations are used throughout the section:

- RST abbreviates `<wst:RequestSecurityToken>`.
- RSTR abbreviates `<wst:RequestSecurityTokenResponse>`.

The following sub-sections specify the individual and mandatory SOAP body elements and attributes within the `wst:RequestSecurityToken` element.

2.3.2.1 @Context

The `Context` attribute is required.

The `Context` attribute uniquely and opaquely identifies the RST and is used for correlating the RSTR to this. The value of `Context` must be an opaque URI that must comply with the following convention:

”urn:uuid:[unique identifier]”

```
<wst:RequestSecurityToken Context="urn:uuid:eb529fec-6498-11d7-b236-
000629ba5445">
```

WS-Trust [WST] does not define any requirements for creating the unique identifier. It is recommended that the unique identifier follows the [xs:ID] simple type.

General processing rules:

- WSC is responsible for creating unique `Context` identifiers (STS does not verify uniqueness).
- The `Context` identifier must not be identical with any other identifiers created for the SOAP header or body block.

2.3.2.2 /wst:TokenType

The element is optional.

The element specifies the type of token that will be returned in RSTR.

STS currently supports SAML V2.0 assertions only and any other token URI references must cause STS to reject the request.

Accepted token URI reference is: http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0

```
<wst:TokenType>
  http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0
</wst:TokenType>
```

2.3.2.3 /wst:RequestType

The element is required.

STS supports only the issuance binding and the element must therefore use the following URI: http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue.

```
<wst:RequestType>
  http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue
</wst:RequestType>
```

2.3.2.4 /wst14:ActAs

The element is required for bootstrap token case and local token case scenarios.

The element must not be present in signature case scenario. The request must be rejected by STS if present.

The element must contain a bootstrap token that represents the user the request is made on behalf-of. The token must be embedded directly and thus not be a token reference.

```
<wst14:ActAs>
  <saml:Assertion>
    <!-- token for user that is acted on behalf of here -->
  </saml:Assertion>
</wst14:ActAs>
```

General bootstrap processing rules:

- The bootstrap token must not be encrypted.
- The bootstrap token is allowed to be Base64 encoded but not required
- The bootstrap token must be signed by issuer.

The bootstrap token must comply with formatting and rules defined in the subsequent sections.

2.3.2.4.1 /saml2:Assertion/@ID

The attribute is required.

The `ID` attribute must contain the identifier of the assertion and must be of the type [xs:ID].

General processing rules:

- WSC is responsible for creating unique `ID` identifiers (STS does not verify uniqueness).
- The `ID` identifier must not be identical with the any other identifiers created for the SOAP header or body block.

2.3.2.4.2 /saml2:Assertion/@IssueInstant

The attribute is required.

The `IssueInstant` attribute must contain the assertion time instant of issue in UTC time specified by the XML Schema type [xs:dateTime] (YYYY-MM-DDThh:mm:ss.sZ).

```
IssueInstant="2013-09-05T11:57:52.2375322Z"
```

General processing rules:

- `IssueInstant` must be before STS current time and should not differ substantially from the SOAP request transmission time, which must be governed by STS.

If time instant rules are not met the SOAP request is rejected.

2.3.2.4.3 /saml2:Assertion/@Version

The attribute is required.

The `Version` attribute must hold the SAML version of the assertion. The identifier for the version of SAML defined in this specification is "2.0".

2.3.2.4.4 /saml2:Assertion/saml2:Issuer

The element is required.

The `Issuer` element must contain the entityId of the party issuing the assertion.

General processing rules:

- The issuing party must be the entityId of Nemlog-in Web SSO or the entityId of a trusted local STS (that are registered in Nemlog-in CSS).

2.3.2.4.5 /saml2:Assertion/ds:Signature

The element is required.

The `Signature` element contains an XML Signature that protects the integrity of and authenticates the issuer of the assertion. The signature must follow the XML Signature profile described in [SAML-CORE] and must cover the complete assertion.

The signature must follow the normative example shown below:

```
<!-- signature by the issuer over the assertion -->
    <ds:Signature>
      <ds:SignedInfo>
        <ds:CanonicalizationMethod Algorithm="..."/>
        <ds:SignatureMethod Algorithm="..."/>
        <!-- References the assertion ID attribute -->
        <ds:Reference URI="...">
          <ds:Transforms>
            <ds:Transform Algorithm="..."/>
          </ds:Transforms>
          <ds:DigestMethod Algorithm="..."/>
          <ds:DigestValue>
            <!-- digest value -->
          </ds:DigestValue>
        </ds:Reference>
      </ds:SignedInfo>
      <ds:SignatureValue>
        <!-- Signature value -->
      </ds:SignatureValue>
   </ds:Signature>
```

If the signature element contains a `/ds:KeyInfo/ds:X509Data/ds:X509Certificate` element this is ignored by STS. STS uses the Nemlog-in signing certificate or a local STS signing certificate registered in Nemlog-in CSS to verify signature, depending on usage scenario. Pls. refer the "Usage scenario processing rules" below.

The following signature algorithms are supported by STS:

Signature element	Algorithm
`CanonicalizationMethod`	http://www.w3.org/2001/10/xml-exc-c14n#
`SignatureMethod`	http://www.w3.org/2000/09/xmldsig#rsa-sha1 http://www.w3.org/2001/04/xmldsig-more#rsa-sha256
`Transforms`	http://www.w3.org/2000/09/xmldsig#enveloped-signature
`DigestMethod`	http://www.w3.org/2000/09/xmldsig#sha1 http://www.w3.org/2001/04/xmlenc#sha256

General signature processing rules:

The bootstrap token signing certificate must obey with the general rules below:

- The signing certificate must be a VOCES or FOCES certificate depending on the usage scenario (see "Usage scenario processing rules" below).
- The signing certificate validity period must be valid.
- The signing certificate must not be revoked.
- The signing certificate must be issued directly by a DanID CA certificate (any certificates between the root certificate and the signing certificate must be DanID CA certificates).
- The DanID CA certificate(s) must not be revoked.

Usage scenario processing rules:

In addition to the general signature processing rules, the signature must obey with rules below depending on usage scenario:

Usage scenario	Processing rule
Bootstrap token case	Signature must verify with Nemlog-in's signing certificate.
Local token case	Signature must verify with a VOCES or FOCES certificate that is registered as trusted local STS in Nemlog-in CSS.

The entityId contained in the `/saml2:Assertion/saml2:Issuer` element is used to identify the expected signing certificate.

2.3.2.4.6 /saml2:Assertion/saml2:Subject

The element is required.

The `Subject` element identifies the user the request is made on behalf-of.

The subject must conform to the normative example below:

```
<saml2:Subject>
  <saml2:NameID Format="...">
    <!-- Identifies the subject -->
  </saml2:NameID>
   <saml2:SubjectConfirmation Method="...">
   </saml2:SubjectConfirmation>
</saml2:Subject>
```

2.3.2.4.6.1 /saml2:Assertion/saml2:Subject/saml2:NameID

The `Subject` element must contain a `NameID` element that identifies the subject (user). The following formats are supported:

- `urn:oasis:names:tc:SAML:2.0:nameid-format:persistent`

- `urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName`

The `NameID` identifier must comply with the rules defined below, which depends on usage scenario and `NameID` format:

Usage scenario	X509SubjectName	Persistent
Bootstrap token case	• WSC must present a correctly formatted POCES or MOCES X509SubjectName (user certificate distinguished name). • The formatting must comply with "Requirements for the Subject Element" in [OIO-WEB-SSO].	• WSC must present a persistent pseudonym identifier (hereafter named WSC identifier) that is issued by Nemlog-in Web SSO in accordance with the Persistent Pseudonym Attribute Profile [OIO-WEB-SSO]. • The WSC identifier is a shared "secret" between WSC (SP) and Nemlog-in Web SSO/STS.
Local token case	• WSC must present a correctly formatted MOCES X509SubjectName (user certificate distinguished name). • The formatting must comply with "Requirements for the Subject Element" in [OIO-WEB-SSO]. • The X509SubjectName must belong to the local STS organization – must be bound to the same CVR-number as the certificate used for signing the embedded bootstrap token. • POCES certificates are not accepted.	• WSC must present an identifier (hereafter named WSC identifier) that is issued by the local STS. • Nemlog-in STS' is not familiar with the user identity behind the WSC identifier.

The WSP the request is requested for accepts either the persistent or X509SubjectName NameID format.

The NameID may in some situations require conversion between WSC and WSP because of the following conditions:

- The NameID format in the WSC request and the NameID format the WSP accepts may differ.

- The NameID format implicates a "secrecy" between parties that cannot be revealed to other parties (e.g. persistent pseudonym identifiers issued by Nemlog-in Web SSO)

The conversion rules are documented in the tables below for the bootstrap token case and the local token case scenarios, respectively.

Bootstrap token case conversion rules:

WSC Request	WSP accepts	NameID conversion rules
Persistent	Persistent	• STS must exchange the WSC identifier to an identifier destined for WSP. STS must maintain mapping between the identifiers for future exchanges. • STS must set the `/saml2:Subject/saml2:NameID` to the WSP identifier in the identity token returned to WSC. Pls. refer section 2.5 "Response message" for details.
Persistent	X509-SubjectName	• STS must translate the WSC identifier to the corresponding X509Subjectname (user certificate distinguished name). STS must reject the request if translation is not possible. • STS must set the `/saml2:Subject/saml2:NameID` to the translated X509SubjectName in the identity token returned to WSC. Pls. refer section 2.5 "Response message" for details.
X509-SubjectName	Persistent	• STS must exchange the X509SubjectName to a persistent pseudonym identifier destined for WSP. STS must maintain mapping between X509SubjectName and the WSP identifier for future exchanges. • STS must set the `/saml2:Subject/saml2:NameID` to the WSP identifier in the identity token returned to WSC. Pls. refer section 2.5 "Response message" for details.
X509-SubjectName	X509-SubjectName	• STS must set the `/saml2:Subject/saml2:NameID` to the X509SubjectName in the identity token returned to WSC. Pls. refer section 2.5 "Response message" for details.

Local token case conversion rules:

WSC Request	WSP Accepts	NameID conversion rules
Persistent	Persistent	• STS must copy the `/saml2:Subject/saml2:NameID` to the WSC identifier in the identity token returned to WSC. Pls. refer section 2.5 "Response message" for details.
Persistent	X509-SubjectName	N/a. • STS must reject the request as STS does not know the associated X509SubjectName.
X509-SubjectName (see table note 1 below)	Persistent	• STS must exchange the X509SubjectName to a persistent pseudonym identifier destined for WSP. STS must maintain mapping between the X509SubjectName and the WSP identifier for future exchanges. • STS must set the `/saml2:Subject/saml2:NameID` to the WSP identifier in the identity token returned to WSC. Pls. refer section 2.5 "Response message" for details.
X509-SubjectName (see table note 1 below)	X509-SubjectName	• STS must copy the `/saml2:Subject/saml2:NameID` to the X509SubjectName in the identity token returned to WSC. Pls. refer section 2.5 "Response message" for details.

2.3.2.4.6.2 /saml2:Assertion/saml2:Subject/saml2:SubjectConfirmation

The element is required.

The `SubjectConfirmation` element contains information that instructs how the `Subject` is verified and confirmed.

Bootstrap token case processing rules

- The `Subject` element must contain a `SubjectConfirmation` element that must reference the `urn:oasis:names:tc:SAML:2.0:cm:holder-of-key` method.
- The `SubjectConfirmation` element must contain a `SubjectConfirmationData` element that must contain the certificate used for signing the SOAP request.
- The `SubjectConfirmationData` element must contain a `KeyInfoConfirmationDataType` attribute.
- If the `SubjectConfirmationData` element contains time instants they must fall within the overall assertion validity period as specified by the `Conditions` element's `NotBefore` and `NotOnOrAfter` attributes. If both attributes are present, the value for `NotBefore` MUST be less than (earlier than) the value for `NotOnOrAfter`.

`Subject` formatting example:

```
<saml2:Subject>
  <saml2:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName">
    C=DK,O=ACME A/S // CVR:11111111,CN=Tola Kristiansen,Serial=CVR:11111111-RID:48245447
  </saml2:NameID>
   <saml2:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:holder-of-key">
      <saml2:SubjectConfirmationData xsi:type="saml2:KeyInfoConfirmationDataType">
          <ds:KeyInfo>
            <ds:X509Data>
              <!-- X509 cert used for signing SOAP request -->
            </ds:X509Data>
          </ds:KeyInfo>
        </saml2:SubjectConfirmationData>
   </saml2:SubjectConfirmation>
</saml2:Subject>
```

Local token case processing rules (“holder-of-key”)

- The `Subject` element must contain a `SubjectConfirmation` element that must reference the `urn:oasis:names:tc:SAML:2.0:cm:holder-of-key` method.
- The `SubjectConfirmation` element must contain a `SubjectConfirmationData` element that must contain the certificate used for signing the SOAP request.
- The `SubjectConfirmationData` element must contain a `KeyInfoConfirmationDataType` attribute.
- If the `SubjectConfirmationData` element contains time instants they must fall within the overall assertion validity period as specified by the `Conditions` element's `NotBefore` and `NotOnOrAfter` attributes. If both attributes are present, the value for `NotBefore` MUST be less than (earlier than) the value for `NotOnOrAfter`.

`Subject` formatting example:

```
<saml2:Subject>
  <saml2:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName">
    C=DK,O=ACME A/S // CVR:11111111,CN=Tola Kristiansen,Serial=CVR:11111111-RID:48245447
  </saml2:NameID>
   <saml2:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:holder-of-key">
      <saml2:SubjectConfirmationData xsi:type="saml2:KeyInfoConfirmationDataType>"
          <ds:KeyInfo>
            <ds:X509Data>
              <!-- X509 cert used for signing SOAP request -->
            </ds:X509Data>
          </ds:KeyInfo>
        </saml2:SubjectConfirmationData>
   </saml2:SubjectConfirmation>
</saml2:Subject>
```

Local token case processing rules (“bearer”)

- The `Subject` element must contain a `SubjectConfirmation` element that must reference the `urn:oasis:names:tc:SAML:2.0:cm:bearer` method.
- The `SubjectConfirmation` element may contain a `SubjectConfirmationData` element, which must be ignored by STS.

`Subject` formatting example:

```
<saml2:Subject>
  <saml2:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName">
    C=DK,O=ACME A/S // CVR:11111111,CN=Tola Kristiansen,Serial=CVR:11111111-RID:48245447
  </saml2:NameID>
   <saml2:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
    </saml2:SubjectConfirmation>
</saml2:Subject>
```

2.3.2.4.7 /saml2:Assertion/saml2:Conditions

The element is required.

The `Conditions` element must contain conditions that must be met before accepting the assertion.

The `Conditions` element must contain a `NotOnOrAfter` attribute that specifies a time instant at which the assertion expires. The `Conditions` element may contain a `NotBefore` attribute that specifies a time instant from which the assertion is valid. All time instants must be in UTC format as specified by the XML Schema type [xs:dateTime] (YYYY-MM-DDThh:mm:ss.sZ).

Time instant processing rules:

- `NotOnOrAfter` must be later than `NotBefore` and before current STS time.
- `NotBefore` should not differ substantially from the SOAP request transmission time, which must be governed by STS.
- If time instants are not met the SOAP request is rejected.

Time instants formatting example:

```
<Conditions NotBefore="2012-02-23T09:28:45.890Z" NotOnOrAfter="2012-02-23T10:28:45.890Z">
…
</Conditions>
```

2.3.2.4.7.1 /saml2:Assertion/saml2:Conditions/saml2:AudienceRestriction

The element is required.

The `Conditions` element must contain an `AudienceRestriction` element that again must contain an `Audience` element. The `Audience` element must contain the entityId of STS and the entityId must be the same as specified in section 2.3.1.3 "/wsa:To".

`Audience` element formatting example:

```
<Conditions>
  <AudienceRestriction>
    <Audience>https://bootstrap.sts.nemlog-in.dk</Audience>
  </AudienceRestriction>
</Conditions>
```

2.3.2.4.8 /saml2:Assertion/saml2:AttributeStatement

The element is optional in the local token case scenario.

The element is required in the bootstrap token case scenario, though only the private attribute defined in section 2.3.3 "Session index check" and private attribute "dk:nemlogin:saml:attribute:SpEntityId" is accepted.

The `AttributeStatement` element must contain statements (attributes) about the user specified in subject. The attributes are transferred to the identity token in accordance with the rules defined in the section 2.4 "Attribute filtering".

The `AttributeStatement` element must conform to the normative example below:

```
<saml2:AttributeStatement>
  <saml2:Attribute Name="..." NameFormat="...">
    <saml2:AttributeValue>...</saml2:AttributeValue>
  </saml2:Attribute>
  <!-- More attribute statements -->
 </saml2:AttributeStatement>
```

The `AttributeStatement` element must contain one or more `Attribute` elements that each represents statements (attributes and attribute values) about the user. An `Attribute` element must contain a `Name` attribute that uniquely identifies the attribute. Pls. refer the section 2.4 "Attribute filtering" for lists of acceptable attribute identifiers. An `Attribute` element must contain a `NameFormat` attribute that must be set to `urn:oasis:names:tc:SAML:2.0:attrname-format:basic`

An `Attribute` element must contain one or more `AttributeValue` elements that must contain the attribute value(s). An `AttributeValue` element may contain a `xsi:type` declaration that declares the attribute value data type explicitly using the XML Schema simple type. An `AttributeValue` element without data type declaration will automatically default to string (xs:string type).

The `Attribute` element formatting example:

```
<saml2:Attribute
  Name="dk:gov:saml:attribute:RidNumberIdentifier"
  NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
  <saml2:AttributeValue>Some value</saml2:AttributeValue>
</saml2:Attribute>
```

2.3.2.5 /wsp:AppliesTo

The element is required.

The `AppliesTo` element must contain exactly one `wsa:EndpointReference` that must contain an entityId that identifies the WSP for which the identity token should be issued.

`AppliesTo` element formatting and normative example:

```
<wsp:AppliesTo>
  <wsa:EndpointReference>
    <wsa:Address>https://wsp.someorganization.dk</wsa:Address>
  </wsa:EndpointReference>
</wsp:AppliesTo>
```

Endpoint processing rules:

- The `EndpointReference` entityId must be registered as WSP in Nemlog-in CSS.

2.3.2.6 /wst:Lifetime

The element is optional.

The `LifeTime` element may be used to indicate the desired valid time range of the identity token to be issued. STS is not obligated to honor this range and may return a token with a shorter life time in RSTR.

The `LifeTime` element may contain one `wsu:Created` element and must contain one `wsu:Expires` element. All time instants must be in UTC format as specified by the XML Schema type [xs:dateTime] (YYYY-MM-DDThh:mm:ssZ).

`LifeTime` element formatting and normative example:

```
<wst:Lifetime>
  <wsu:Created>2004-05-06T22:04:34Z</wsu:Created>
  <wsu:Expires>2004-05-07T10:04:34Z</wsu:Expires>
</wst:Lifetime>
```

Time instants processing rules:

- `Expires` must be accepted if this falls within the allowed identity token life time policy[1].
- `Created` must be ignored by STS (identity token valid from must always be set to identity token lifetime policy[1] defaults).
- If the above rules are not met, STS must adjust RSTR to identity token lifetime policy[1] defaults. STS returns a `Lifetime` element in the RSTR, which is always the normative lifetime.
- If the `Lifetime` element is not included in RST, STS returns a `Lifetime` element in the RSTR, which is always the normative lifetime.

[1]Pls. refer section 2.8 "Identity token lifetime policy"

2.3.3 Session index check

The session index check is a verification of whether the user a request is made on behalf-of, is still having an active user session with Nemlog-in Web SSO.

The session index check is only relevant in the bootstrap token case and only if the WSP, for which the requested identity token is requested for, requires that a session index check is performed.

The session index associated with a user session must be present in the bootstrap token through the private attribute `dk:nemlogin:saml:attribute:IdPSessionIndex`.

Session index verification processing rules:

- If the private attribute is not available or its attribute value is empty, the request is rejected.
- If the session index is not known by Nemlog-in Web SSO or the associated user session has expired, the request is rejected.

2.4 Attribute filtering

This section specifies which attributes that are supported and the attribute filtering processing rules.

The STS supports two attribute profiles as specified in OIOSAML [OIO-WEB-SSO]:

- OCES attribute profile
- Persistent Pseudonym Attribute Profile

Rules for determine the attribute profile to effectuate:

- The OCES attribute profile is effectuated when the WSP, the identity token is requested for, accepts X509SubjectName subject identifiers.
- The Persistent Pseudonym Attribute Profile is effectuated when the WSP, the identity token is requested for, accepts persistent subject identifiers.

Pls. refer the section 2.3.2.4.6.1 "/saml2:Assertion/saml2:Subject/saml2:NameID" for details on WSP subject identifier acceptance.

2.4.1 Attributes

2.4.1.1 OCES attribute profile

The following table specifies the complete list of attributes that can be returned by the STS for the OCES attribute profile. The actual set of attributes returned (within this list) is defined by the WSP for which the identity token is requested for.

Friendly name	Attribute name
SpecVer	dk:gov:saml:attribute:SpecVer
AssuranceLevel	dk:gov:saml:attribute:AssuranceLevel

Friendly name	Attribute name
Surname	urn:oid:2.5.4.4
CommonName	urn:oid:2.5.4.3
Uid	urn:oid:0.9.2342.19200300.100.1.1
Mail	urn:oid:0.9.2342.19200300.100.1.3
serialNumber	urn:oid:2.5.4.5
organizationName[2)]	urn:oid:2.5.4.10
IsYouthCert[1)]	dk:gov:saml:attribute:IsYouthCert
userCertificate	urn:oid:1.3.6.1.4.1.1466.115.121.1.8
Certificate issuer attribute	urn:oid:2.5.29.29
ProductionUnitIdentifier[2)]	dk:gov:saml:attribute:ProductionUnitIdentifier
UserAdministratorIndicator[2)]	dk:gov:saml:attribute:UserAdministratorIndicator
SeNumberIndentifier[2)]	dk:gov:saml:attribute:SENumberIdentifier
CprNumberIdentifier	dk:gov:saml:attribute:CprNumberIdentifier
PidNumberIdentifier[1)]	dk:gov:saml:attribute:PidNumberIdentifier
CVRnumberIdentifier[2)]	dk:gov:saml:attribute:CvrNumberIdentifier
RidNumberIdentifier[2)]	dk:gov:saml:attribute:RidNumberIdentifier
UniqueAccountKey	dk:gov:saml:attribute:UniqueAccountKey
Postal address [3)]	urn:oid:2.5.4.16
Title[3)]	urn:oid:2.5.4.12
Organization unit[2) 3)]	urn:oid:2.5.4.11
Privileges	dk:gov:saml:attribute:Privileges_intermediate

The attribute footnote references in the table refer to the following:

1) Attributes that is specific to citizen certificate, thus their value can only be returned when the request is send on-behalf of a POCES certificate.
2) Attributes that is specific to employee certificate, thus their value can only be returned when the request is send on-behalf of a MOCES certificate.
3) Attributes that are specified in the OIOSAML profile [OIO-WEB-SSO] and accepted to be requested by STS. However, STS does not know the corresponding value, so their value will not be returned[4].

[4] The attribute values will be returned if contained in the bootstrap token in the local token case. Pls. refer section 2.4.2.2 for details on this.

Requested attributes that doesn't have a value will still be returned by STS in accordance with the rules defined in section 2.5.2.4.11 "Saml2:Assertion/Saml2:AttributeStatement".

2.4.1.2 Persistent Pseudonym Attribute Profile

The following table specifies the complete list of attributes that can be returned by the STS for the Persistent Pseudonym attribute profile.

Friendly name	Attribute name
SpecVer	dk:gov:saml:attribute:SpecVer
AssuranceLevel	dk:gov:saml:attribute:AssuranceLevel
IsYouthCert[1)]	dk:gov:saml:attribute:IsYouthCert
Certificate issuer attribute	urn:oid:2.5.29.29

The attribute footnote references in the table refer to the following:

1) Attributes that is specific to citizen certificate, thus their value can only be returned when the request is send on-behalf of a POCES certificate.

2.4.2 Processing rules

This section specifies requirements related to processing of attributes:

- The general processing rule is that the STS returns the attributes that the WSP has specified in metadata registered in Nemlog-in CSS. The attribute values are collected from STS attribute stores such as FBRS and certificate cache.
- However, for the Local token case the bootstrap token may contain attributes. If attributes are present in the bootstrap token then they must take precedence over the values returned by Nemlog-in, if they comply with the rules defined in section 2.4.2.2 "Processing rule for local token case". With respect to the privilege attribute then the identity token must contain the union of the privileges from the bootstrap token and FBRS. For the other attributes the values in the bootstrap token overrides the values in STS.

2.4.2.1 General processing rule

- The attributes specified in the WSP's metadata must be included in the token
- The attribute values are retrieved from STS attribute stores

2.4.2.2 Processing rule for local token case

The following rules must be applied if the bootstrap token contains one or more attributes:

- All bootstrap attributes are copied to the identity token. This also includes attributes not specified by the WSP metadata.
- If any of the bootstrap attributes are a known KFOBS Oces attribute the attribute name must be correctly cased or the request will be rejected.
- If the bootstrap token contains OIOSAML attributes that are not allowed for the attribute profile (OCES attribute or persistent pseudonym profile) then these attributes are removed from the response. For example, if the bootstrap token contains the dk:gov:saml:attribute:CvrNumberIdentifier attribute and the attribute profile is persistent pseudonym then the attribute will not be included in the response identify token.
- Bootstrap token attributes must be type declared using the XML Schema simple type, such as xs:integer, xs:string etc. The request is rejected if this rule is violated.
- The bootstrap token must include the AssuranceLevel attribute. The request is rejected if this rule is violated.
- If the WSP metadata specify attributes that are not included in the bootstrap token then STS adds these attributes to the identity token.
- If the bootstrap token contains cvrNumberIdentifier then it must be the CVR of the WSC's organization. The request is rejected if this rule is violated.
- If the bootstrap token contains ProductionUnitIdentifier then it must belong to the WSC's organization. The request is rejected if this rule is violated.
- If the bootstrap token contains SeNumberIndentifier then it must belong to the WSC's organization. The request is rejected if this rule is violated.
- The bootstrap token must only contain privileges defined by the WSP. The request is rejected if this rule is violated.
- The privileges in the bootstrap token must have a scope within the WSC's organization (CVR, production unit or SE-unit). Citizen delegation privileges are not allowed. The request is rejected if this rule is violated.
- Privileges retrieved from FBRS will be joined with the privileges in the bootstrap token.

2.5 Response message

This section specifies requirements related to response message interface, processing and formatting, which STS and subsequent WSC's must adhere to.

Response message formatting

A response message must contain the elements shown below:

```
<S11:Envelope>
  <S11:Header>
    <wsa:Action>…</wsa:Action>
    <wsa:MessageID>…</wsa:MessageID>
    <wsa:RelatesTo>…</wsa:RelatesTo>
    <wsse:Security>…</wsse:Security>
  </S11:Header>
  <S11:Body>
    <wst:RequestSecurityTokenResponseCollection>
      <wst:RequestSecurityTokenResponse>
        <wst:TokenType>…</wst:TokenType>
        <wsp:AppliesTo>…</wsp:AppliesTo>
        <wst:RequestedSecurityToken>
           <saml2:EncryptedAssertion>…</saml2:EncryptedAssertion>
        </wst:RequestedSecurityToken>
        </wst:LifeTime>…</wst:LifeTime>
      </wst:RequestSecurityTokenResponse>
    </wst:RequestSecurityTokenResponseCollection>
  </S11:Body>
</S11:Envelope>
```

2.5.1 SOAP header

This section specifies requirements related to the SOAP response header block.

2.5.1.1 /wsa:Action

The element is always returned by STS.

The `Action` echoes the `Action` URI identifier specified in the RST.

2.5.1.2 /wsa:MessageID

The element is always returned by STS.

The `MessageID` element is set to a uniquely and opaquely identifier that identifies the SOAP response using the convention ”uuid:”+*[unique identifier]*, where the unique identifier follows the [xs:ID] simple type

`MessageID` element formatting example:

```
<wsa:MessageID>
  uuid:aaaabbbb-cccc-dddd-eeee-ffffffffffff
</wsa:MessageID>
```

2.5.1.3 /wsa:RelatesTo

The element is always returned by STS.

The `RelatesTo` echoes the `MessageID` identifier specified in the RST.

2.5.1.4 /wsse:Security

The element is always returned by STS.

The `Security` element contains the signature over the SOAP response. The SOAP response is signed using STS signing certificate.

The signature is issued in accordance with the normative example below:

```
<wsse:Security mustUnderstand="1">
   <wsu:Timestamp>…</wsu:Timestamp>
   <ds:Signature>
     <ds:SignedInfo>
       <ds:CanonicalizationMethod Algorithm="…"/>
       <ds:SignatureMethod Algorithm="…"/>
         <!—references to elements included in signature →
       <ds:Reference URI="#action">…</ds:Reference>
       <ds:Reference URI="#messageid">…</ds:Reference>
       <ds:Reference URI="#relatesto">…</ds:Reference>
       <ds:Reference URI="#sec_timestamp">…</ds:Reference>
       <ds:Reference URI="#body">…</ds:Reference>
     </ds:SignedInfo>
     <ds:SignatureValue>…</ds:SignatureValue>
  </ds:Signature></wsse:Security>
```

The `Security@mustUnderstand` attribute is set to true.

The `Timestamp` element is returned with both the `wsu:Created` and the `wsu:Expires` elements. The time instants are identical to the wst:LifeTime time instants.

`Timestamp` element normative example:

```
<wsu:Timestamp wsu:Id="…">
  <wsu:Created>2004-05-06T22:04:34Z</wsu:Created>
  <wsu:Expires>2004-05-06T22:04:34Z</wsu:Expires>
</wsu:Timestamp>
```

The signature covers all response message elements using `ds:Reference` for each covered element.

STS uses the following signature algorithms:

Signature element	Algorithm
CanonicalizationMethod	http://www.w3.org/2001/10/xml-exc-c14n#
SignatureMethod	http://www.w3.org/2000/09/xmldsig#rsa-sha1
Transforms	http://www.w3.org/2000/09/xmldsig#envelopedsignature
DigestMethod	http://www.w3.org/2000/09/xmldsig#sha1

2.5.2 SOAP body

This section specifies requirements related to the SOAP response body block

General processing rules:

- SOAP body must contain exactly one `wst:RequestSecurityTokenResponseCollection` element with exactly one `wst:RequestSecurityTokenResponse`.

The following abbreviations are used throughout the section:

- RST abbreviates `wst:RequestSecurityToken`.
- RSTR abbreviates `wst:RequestSecurityTokenResponse`.

The following sub-sections specify the individual and mandatory SOAP body elements and attributes within the `wst:RequestSecurityTokenResponse` element.

2.5.2.1 @Context

The attribute is always returned by STS.

The `Context` echoes the `context` identifier specified in the RST.

2.5.2.2 /wst:TokenType

The element is always returned by STS.

The `TokenType` specifies the type of identity token returned in the RSTR. The `TokenType` echoes the `TokenType` URI specified in the RST.

As STS currently supports SAML V2.0 assertions only, STS will always return the URI associated with this: http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0

2.5.2.3 /wsp:AppliesTo

The element is always returned by STS.

The `AppliesTo` specifies the entityId of the WSP the identity token is requested for. The `AppliesTo` echoes the `AppliesTo` entityId specified in the RST.

2.5.2.4 /wst:RequestedSecurityToken

The element is always returned by STS.

The `RequestedSecurityToken` contains the identity token destined for the WSP specified in the `AppliesTo` element:

- The identity token is represented by a SAML assertion in encrypted fashion, (`saml2:EncryptedAssertion`) as defined by XML encryption [XML-ENC]. The identity token is encrypted using WSP's public key, which must be pre-registered with Nemlog-in CSS.

- The `saml2:EncryptedAssertion` element is directly embedded in the `RequestedSecurityToken` element.
- The `saml2:EncryptedAssertion` element contains a `xenc:EncryptedData` element that contains the encrypted content and associated encryption details.
- The `xenc:EncryptedData` element contains an `wsu:id` attribute that enables the requestor (WSC) to reference the identity token.
- The `xenc:EncryptedData` element contains a `xenc:EncryptedKey` element that wraps the encrypted key used for decrypting the assertion.

STS uses the following encryption methods:

- aes256-cbc encryption method for block encryption.
- rsa-oaep-mgf1p encryption method for key transport.

The `saml2:EncryptedAssertion` is formatted according to the normative example below:

```
<wst:RequestedSecurityToken>
  <saml2:EncryptedAssertion xmlns:saml2=”urn:oasis:names:tc:SAML:2.0:assertion”>
    <xenc:EncryptedData id=”…” Type=”http://www.w3.org/2001/04/xmlenc#Element”
                         xmlns:xenc=”http://www.w3.org/2001/04/xmlenc#”>
      <xenc:EncryptionMethod Algorithm=”http://www.w3.org/2001/04/xmlenc#aes256-cbc” />
      <KeyInfo xmlns=”http://www.w3.org/2000/09/xmldsig#”>
        <e:EncryptedKey xmlns:e=”http://www.w3.org/2001/04/xmlenc#”>
          <e:EncryptionMethod Algorithm=”http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p”>
            <DigestMethod Algorithm=”http://www.w3.org/2000/09/xmldsig#sha1” />
          </e:EncryptionMethod>
          <!–STS may add a
              <ds:KeyInfo>, <ds:X509Data>, <ds:X509IssuerSerial> or <ds:X509SubjectName>
           reference to the WSP certificate used for encryption →
          <e:CipherData>
            <e:CipherValue>
              <!–encryption key encrypted using WSP’s public key →
            </e:CipherValue>
          </e:CipherData>
        </e:EncryptedKey>
      </KeyInfo>
      <xenc:CipherData>
        <xenc:CipherValue>
          <!–encrypted SAML assertion →
        </xenc:CipherValue>
      </xenc:CipherData>
    </xenc:EncryptedData>
  </saml2:EncryptedAssertion>
</wst:RequestedSecurityToken>
```

2.5.2.4.1 /saml2:Assertion

This section specifies how assertions are processed and formatted by STS prior to encrypting assertions.

2.5.2.4.2 /saml2:Assertion/@ID

The `ID` attribute is set to a unique identifier of the type [xs:ID].

2.5.2.4.3 /saml2:Assertion/@IssueInstant

The `IssueInstant` attribute is set to the time instant of the `NotBefore` time instant of the `/saml2:Assertion/saml2:Conditions` element.

The `IssueInstant` time instant is issued in UTC time as specified by the XML Schema type [xs:dateTime].

2.5.2.4.4 /saml2:Assertion/@Version

The `Version` attribute is set to the identifier for the version of SAML used by this specification, which is "2.0".

2.5.2.4.5 /saml2:Assertion/saml2:Issuer

The `Issuer` element is set to the usage scenario specific entityId of STS. The `Issuer` echoes the `/wsaTo` entityId specified in the RST.

2.5.2.4.6 /saml2:Assertion/ds:Signature

The `Signature` element contains the signature over the assertion. The assertion is signed using STS' signing certificate.

The signature is issued according to the normative example below:

```
<!–signature by STS over the assertion →
    <ds:Signature>
      <ds:SignedInfo>
        <ds:CanonicalizationMethod Algorithm=”…”/>
        <ds:SignatureMethod Algorithm=”…”/>
        <!–References the assertion ID attribute →
        <ds:Reference URI=”…”>
          <ds:Transforms>
            <ds:Transform Algorithm=”…”/>
          </ds:Transforms>
          <ds:DigestMethod Algorithm=”…”/>
          <ds:DigestValue>
            <!–digest value →
          </ds:DigestValue>
        </ds:Reference>
      </ds:SignedInfo>
      <ds:SignatureValue>
        <!–Signature value →
      </ds:SignatureValue>
   </ds:Signature>
```

STS uses the following algorithms:

Signature element	Algorithm
CanonicalizationMethod	http://www.w3.org/2001/10/xml-exc-c14n#
SignatureMethod	http://www.w3.org/2000/09/xmldsig#rsa-sha1
Transforms	http://www.w3.org/2000/09/xmldsig#envelopedsignature
DigestMethod	http://www.w3.org/2000/09/xmldsig#sha1

2.5.2.4.7 /saml2:Assertion/saml2:Subject/saml2:NameID

The NameID format attribute and NameID identifier is set according to the rules defined in the table below:

Usage scenario	NameID format and identifier
Bootstrap token case	• The NameID format and identifier is determined by the bootstrap token NameID rules defined in section 2.3.2.4.6.1.
Local token case	• The NameID format and identifier is determined by the local token NameID rules defined in section 2.3.2.4.6.1.
Signature case	• The NameID format is always set to urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName • The NameID identifier is always set to the distinguished name (DN) from the MOCES certificate used to sign the SOAP request.

2.5.2.4.8 /saml2:Assertion/saml2:Subject/saml2:SubjectConfirmation

The SubjectConfirmation method is set according to the rules defined in the table below:

Usage scenario	Subject confirmation method
Bootstrap token case	• SubjectConfirmation method is always set to urn:oasis:names:tc:SAML:2.0:cm:holder-of-key • The certificate used to sign the SOAP request is used as reference (see above).

Usage scenario	Subject confirmation method
Local token case ("holder-of-key")	• `SubjectConfirmation method` is always set to `urn:oasis:names:tc:SAML:2.0:cm:holder-of-key` • The certificate used to sign the SOAP request is used as reference (see above).
Local token case ("bearer")	• `SubjectConfirmation method` is always set to `urn:oasis:names:tc:SAML:2.0:cm:bearer`
Signature case	• `SubjectConfirmation method` is always set to `urn:oasis:names:tc:SAML:2.0:cm:bearer`

If the `SubjectConfirmation method` is set to `urn:oasis:names:tc:SAML:2.0:cm:holder-of-key` STS adds the SOAP request signing certificate as reference in accordance with the example below:

```
<saml2:SubjectConfirmationData>
  <ds:KeyInfo>
     <ds:X509Data>
        <ds:X509Certificate>
           <!–X509 certificate that must sign subsequent request to WSP  →
        </ds:X509Certificate>
     </ds:X509Data>
  </ds:KeyInfo>
</saml2:SubjectConfirmationData>
```

2.5.2.4.9 /saml2:Assertion/saml2:Conditions

The `Conditions` element attributes `NotBefore` and `NotOnOrAfter` is set according to section 2.5.2.5 "wst:LifeTime".

2.5.2.4.10 /saml2:Assertion/saml2:Conditions/saml2:AudienceRestriction

The `AudienceRestriction/saml2:Audience` element is set to the entityId of the WSP the identity token is requested for.

The `Audience` echoes the `AppliesTo` entityId specified in section 2.5.2.3 "/wsp:AppliesTo".

2.5.2.4.11 /saml2:Assertion/saml2:AttributeStatement

The `AttributeStatement` contains the attributes returned in accordance with section 2.4 "Attribute filtering".

- `Attribute` elements are formatted according to [OIO-WEB-SSO] attribute encoding rules and includes:
  - The "FriendlyName" attribute.
  - The "NameFormat" attribute, which is set to "urn:oasis:names:tc:SAML:2.0:attrname-format:basic".

  - The ”Name” attribute.
- `AttributeValue` element data type is declared explicitly using the XML Schema simple type, such as xs:integer or xs:string:
  - STS controlled attributes are type declared with xs:string in accordance with OIOSAML [OIO-WEB-SSO].
  - The type declared for attributes in bootstrap tokens (local token case) are copied together with the attributes.

Attributes and attribute values are issued according to the formatting below:

```
<saml2:Attribute
  FriendlyName=”SurName”
  NameFormat=”urn:oasis:names:tc:SAML:2.0:attrname-format:basic”
  Name=”urn:oid:2.5.4.4”>
  <saml2:AttributeValue xsi:type=”xs:string”>
    Larsen
  </saml2:AttributeValue>
</saml2:Attribute>
```

An attribute that has no known source is returned according to the formatting below:

```
<saml:Attribute
  FriendlyName=”UniqueAccountKey”
  NameFormat=”urn:oasis:names:tc:SAML:2.0:attrname-format:basic”
  Name=”dk:gov:saml:attribute:UniqueAccountKey”>
  <saml:AttributeValue xsi:nil=”true”/>
</saml:Attribute>
```

An attribute value of “Null” is returned according to the formatting below:

```
<saml:Attribute
  FriendlyName=”UniqueAccountKey”
  NameFormat=”urn:oasis:names:tc:SAML:2.0:attrname-format:basic”
  Name=”dk:gov:saml:attribute:UniqueAccountKey”>
  <saml:AttributeValue xsi:nil=”true”/>
</saml:Attribute>
```

An attribute with a known source, but has no value/empty value, is returned according to the formatting below:

```
<saml:Attribute
  FriendlyName=”UniqueAccountKey”
  NameFormat=”urn:oasis:names:tc:SAML:2.0:attrname-format:basic”
  Name=”dk:gov:saml:attribute:UniqueAccountKey”>
  <saml:AttributeValue/>
</saml:Attribute>
```

2.5.2.5 /wst:Lifetime

The element is always returned by STS.

The `LifeTime` element indicates the actual valid time range of the identity token issued with the RSTS. The time range returned may honor the `LifeTime` time range specified in the RST, if any, or adjusted to the identity token lifetime policy govern by STS. Pls. refer section 2.8 "Identity token lifetime policy"

STS returns both the `wsu:Created` and `wsu:Expires` elements.

`LifeTime` element formatting and normative example:

```
<wst:Lifetime>
  <wsu:Created>2004-05-06T22:04:34Z</wsu:Created>
  <wsu:Expires>2004-05-07T10:04:34Z</wsu:Expires>
</wst:Lifetime>
```

The LifeTime time range and the identity token `Saml2:Assertion/saml2:Conditions` time instants are coordinated:

- @NotBefore = wsu:Created
- @NotOnOrAfter = wsu:Expires

2.6 Response message (fault handling)

STS uses SOAP 1.1 fault mechanism in accordance with [WST] for returning request processing errors.

STS always return the SOAP header elements `wsa:MessageID` and `wsa:RelatesTo` in fault handling, to preserve tracebility between request and response.

Errors are returned and formatted according to the example below:

```
<S11:Envelope>
  <S11:Header>
    <wsa:MessageID>…</wsa:MessageID>
    <wsa:RelatesTo>…</wsa:RelatesTo>
  </S11:Header>
  <S11:Body>
    <S11:Fault>
      <faultcode>
        <!–supported WS-Trust fault code →
      </faultcode>
      <faultstring>
        <!–associated WS-Trust description →
      </faultstring>
    </S11:Fault>
  </S11:Body>
</S11:Envelope>
```

The `fault` element includes the mandatory `faultcode` and `faultstring` elements, which is set in accordance with the [WST] defined fault codes found below.

Fault code	Fault string
wst:InvalidRequest	The request was invalid or malformed
wst:FailedAuthentication	Authentication failed
wst:RequestFailed	The specified request failed
wst:InvalidSecurityToken	Security token has been revoked
wst:BadRequest	The specified RequestSecurityToken is not understood.
Wst:ExpiredData	The request data is out-of-date
wst:InvalidTimeRange	The requested time range is invalid or unsupported

2.7 Audit logging policy

This section specifies requirements related to audit logging capabilities.

STS must audit log the following events and properties:

Log event	Logging properties

STS receives request	The request from WSC must be audit logged with the following information:
	• The UTC time the request is received (format: hh:mm:ss:ms).
	• RemoteIP – requestor (WSP) TCP/IP address.
	• Referrer – requestor (WSP) referrer URL, if any.
	• Request type – indicates the request usage scenario:
	  ○ "Signature case"
	  ○ "Bootstrap token case"
	  ○ "Local token case"
	• Request result – indicates the request rendering result according to the "Request result status codes" below (Request result is for internal use only and must not be revealed in response message).
	• Request message – the complete SOAP envelope

Log event	Logging properties
	contents
STS sends response	The response from STS must be audit logged with the following information: • The UTC time the response is send (format: hh:mm:ss:ms). • Decrypted identity token. • Response message – the complete SOAP envelope contents.

Request result status codes:

Status code	Description
OK	The request is accepted.
Formatting or syntax error	The request doesn't comply with formatting rules defined in this specification.
Request signature error	The SOAP request signature could not be verified according to the rules defined for request type.
Request certificate error	The certificate used for verifying SOAP request signature is not valid due to revocation status or validity.
Bootstrap token signature error	The bootstrap token signature could not be verified according to the rules defined for request type.
Bootstrap token certificate error	The certificate used for verifying bootstrap token signature is not valid due to revocation status or validity.
Unknown WSP error	The WSP indicated as identity token receiver is not trusted by Nemlog-in STS.
NameID conversion error	The NameID could not be converted according to rules defined for NameID conversion.
Attribute filtering error	The rules for filtering attributes could not be satisfied.

2.8 Identity token lifetime policy

The Identity token `/saml2:Conditions` element time instants must be set in accordance with the following:

- `NotBefore` must be set to identity token creation time.

- `NotOnOrAfter` must be set to `NotBefore` + usage scenario token lifetime.

Usage scenario	Token lifetime policy[1]
Bootstrap token case	8 hours
Local token case	8 hours
Signature case	8 hours

Table note 1: adjusted to lifetime requested in `/wst:Lifetime/wsu:Expires` element in SOAP request, if this falls within the token lifetime policy.

3 Normative examples

This section provides normative request, response and token examples for all usage scenarios.

3.1 Bootstrap token case

The message below represents a request destined for STS issued by a WSC (Acme) using the bootstrap token case:

- The request includes a bootstrap token that was issued by Nemlog-in Web SSO to WSC when user authenticated against Nemlog-in Web SSO (the WSC assumed the role as SP during user authentication).
- The bootstrap token contains the `IdpSessionIndex` attribute, which is issued by Nemlog-in Web SSO. The attribute will not be copied to the identity token as the attribute is for internal use by STS only.
- The request is signed by the holder-of-key certificate referenced in the bootstrap token `<saml2:SubjectConfirmationData>` element.
- The bootstrap token is signed by Nemlog-in Web SSO signing certificate.
- The requested identity token is destined for the WSP: https://wsp.someorg.dk

```
<?xml version=”1.0” encoding=”utf-8”?>
<S11:Envelope>
  <S11:Header>
    <wsa:Action wsu:Id=”action”>http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/Issue</wsa:Action>
    <wsa:MessageID wsu:Id=”msgid”>uuid:aaaabbbb-cccc-dddd-eeee-aaaaaaaaaaaa</wsa:MessageID>
    <wsa:To wsu:Id=”to”>https://bootstrap.sts.nemlog-in.dk</wsa:To>
    <wsse:Security mustUnderstand=”1”>
      <wsu:Timestamp wsu:Id=”sec-ts”>
        <wsu:Created>2012-02-23T07:28:45Z</wsu:Created>
        <wsu:Expires>2012-02-23T08:28:45Z</wsu:Expires>
      </wsu:Timestamp>
      <wsse:BinarySecurityToken wsu:Id=”sec-binsectoken”
        ValueType=”http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-
1.0#X509v3”
        EncodingType=”http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-
1.0#Base64Binary”>
        <!–
          X509 certificate used to sign the request →
      </wsse:BinarySecurityToken>
      <ds:Signature>
        <!–
          Signature over request –
          pls. refer request signature example for structure details →
      </ds:Signature>
    </wsse:Security>
  </S11:Header>
  <S11:Body wsu:Id=”body”>
    <wst:RequestSecurityToken Context=”urn:uuid:eb529fec-6498-11d7-b236-000629ba0001”>
      <wst:TokenType>
        http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0
```

```
      </wst:TokenType>
      <wst:RequestType>http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue</wst:RequestType>
      <wst14:ActAs>
        <!–
          The SAML assertion (bootstrap token) issued by Nemlog-in Web SSO →
        <saml2:Assertion ID=”_59ba7540-eaec-4b7a-a12e-d75073f3c001” IssueInstant=”2012-02-
23T07:28:45.578Z”
            Version=”2.0”>
          <saml2:Issuer>https://saml.nemlog-in.dk</saml2:Issuer>
          <ds:Signature>
            <!–
              Signature over assertion using Nemlog-in Web SSO signing certificate
              pls. refer bootstrap token signature example for structure details →
          </ds:Signature>
          <saml2:Subject>
            <saml2:NameID Format=”urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName”>
              C=DK,O=ACME A/S // CVR:11111111,CN=Tola Kristiansen,Serial=CVR:11111111-RID:48245447
            </saml2:NameID>
            <saml2:SubjectConfirmation Method=”urn:oasis:names:tc:SAML:2.0:cm:holder-of-key”>
              <saml2:SubjectConfirmationData xsi:type=”saml2:KeyInfoConfirmationDataType”>
                <ds:KeyInfo>
                  <ds:X509Data>
                    <ds:X509Certificate>
                      <!–
                        X509 certificate that must sign the request to STS  →
                    </ds:X509Certificate>
                  </ds:X509Data>
                </ds:KeyInfo>
              </saml2:SubjectConfirmationData>
            </saml2:SubjectConfirmation>
          </saml2:Subject>
          <saml2:Conditions NotBefore=”2012-02-23T07:28:45.578Z” NotOnOrAfter=”2012-02-
23T08:28:45.578Z”>
            <saml2:AudienceRestriction>
              <saml2:Audience>http://bootstrap.sts.nemlog-in.dk</saml2:Audience>
            </saml2:AudienceRestriction>
          </saml2:Conditions>
          <saml2:AttributeStatement>
            <!–
              The “IdPSessionIndex” attribute is the only allowed in-going attribute
              in the bootstrap token case →
            <saml2:Attribute NameFormat=”urn:oasis:names:tc:SAML:2.0:attrname-format:basic”
              Name=”dk:nemlogin:saml:attribute:IdPSessionIndex”>
              <saml2:AttributeValue xsi:type=”xs:string”>
                <!–
                  User’s session index from Nemlog-in Web SSO →
              </saml2:AttributeValue>
            </saml2:Attribute>
          </saml2:AttributeStatement>
        </saml2:Assertion>
      </wst14:ActAs>
      <wsp:AppliesTo>
        <wsa:EndpointReference>
          <wsa:Address>https://wsp.someorg.dk</wsa:Address>
        </wsa:EndpointReference>
      </wsp:AppliesTo>
      <wst:Lifetime>
        <wsu:Expires>2012-02-23T10:28:45Z</wsu:Expires>
      </wst:Lifetime>
    </wst:RequestSecurityToken>
  </S11:Body>
</S11:Envelope>
```

The message below represents the response issued by STS in reply to the request above:

- The response includes an encrypted SAML assertion (identity token) issued by STS and destined for the WSP.
- The identity token is encrypted using WSP's X509 encrypting certificate and the token can only be decrypted with the corresponding private key.
- The response is signed by STS' signing certificate.

```
<?xml version="1.0" encoding="utf-8"?>
<S11:Envelope>
  <S11:Header>
    <wsa:Action wsu:Id="action">http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/Issue</wsa:Action>
    <wsa:MessageID wsu:Id="msgid">uuid:aaaabbbb-cccc-dddd-eeee-bbbbbbbbbbbb</wsa:MessageID
    <wsa:RelatesTo wsu:Id="relto">uuid:aaaabbbb-cccc-dddd-eeee-aaaaaaaaaaaa</wsa:RelatesTo>
    <wsse:Security mustUnderstand="1">
      <wsu:Timestamp wsu:Id="sec-ts">
        <wsu:Created>2012-02-23T07:28:50Z</wsu:Created>
        <wsu:Expires>2012-02-23T10:28:45Z</wsu:Expires>
      </wsu:Timestamp>
      <ds:Signature>
        <!–
          Signature over response using STS signing certificate –
          pls. refer response signature example for structure details →
      </ds:Signature>
    </wsse:Security>
  </S11:Header>
  <S11:Body wsu:Id="body">
    <wst:RequestSecurityTokenResponseCollection>
      <wst:RequestSecurityTokenResponse Context="urn:uuid:eb529fec-6498-11d7-b236-000629ba0001">
        <wst:TokenType>
          http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0
        </wst:TokenType>
        <wsp:AppliesTo>
          <wsa:EndpointReference>
            <wsa:Address>https://wsp.someorg.dk</wsa:Address>
          </wsa:EndpointReference>
        </wsp:AppliesTo>
        <wst:RequestedSecurityToken>
          <!–
            Encrypted SAML assertion are added here →
          <saml2:EncryptedAssertion>
            <xenc:EncryptedData wsu:Id="enc-token" Type="http://www.w3.org/2001/04/xmlenc#Element">
              <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc" />
              <ds:KeyInfo>
                <xenc:EncryptedKey>
                  <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p">
                    <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
                  </xenc:EncryptionMethod>
                  <xenc:CipherData>
                    <xenc:CipherValue>
                      <!–
                        Encryption key encrypted using WSP's public key →
                    </xenc:CipherValue>
                  </xenc:CipherData>
                </xenc:EncryptedKey>
              </ds:KeyInfo>
              <xenc:CipherData>
                <xenc:CipherValue>
                  <!–
                    Encrypted SAML assertion
                    (The decrypted SAML assertion are documented in separate example) →
                </xenc:CipherValue>
              </xenc:CipherData>
            </xenc:EncryptedData>
```

```
        </saml2:EncryptedAssertion>
      </wst:RequestedSecurityToken>
      <wst:Lifetime>
        <wsu:Created>2012-02-23T07:28:50Z</wsu:Created>
        <wsu:Expires>2012-02-23T10:28:45Z</wsu:Expires>
      </wst:Lifetime>
    </wst:RequestSecurityTokenResponse>
  </wst:RequestSecurityTokenResponseCollection>
 </S11:Body>
</S11:Envelope>
```

The message below represents the decrypted SAML assertion (identity token) included in the response message above:

- The assertion is signed by STS' signing certificate.
- The Subject identifier (user) is the user certificate DN from the request as the WSP in this case accepts X509SubjectName.
- The certificate used for signing the request to STS is referenced as holder-of-key and must be used for signing the subsequent request to the WSP.
- The CVR-number attribute is the only attribute specified in WSP's metadata, hence only this attribute is included.

```
<?xml version="1.0" encoding="utf-8"?>
<saml2:Assertion ID="_59ba7540-eaec-4b7a-a12e-d75073f3c002" IssueInstant="2012-02-23T07:28:50.178Z"
Version="2.0">
  <saml2:Issuer>
    https://bootstrap.sts.nemlog-in.dk
  </saml2:Issuer>
  <ds:Signature>
    <!–
      Signature over assertion using STS signing certificate
      pls. refer assertion signature example for structure details →
  </ds:Signature>
  <saml2:Subject>
    <saml2:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName">
      C=DK,O=ACME A/S // CVR:11111111,CN=Tola Kristiansen,Serial=CVR:11111111-RID:48245447
    </saml2:NameID>
    <saml2:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:holder-of-key">
      <saml2:SubjectConfirmationData xsi:type="saml2:KeyInfoConfirmationDataType">
        <ds:KeyInfo>
          <ds:X509Data>
            <ds:X509Certificate>
              <!–
                X509 certificate that must sign subsequent request to WSP  →
            </ds:X509Certificate>
          </ds:X509Data>
        </ds:KeyInfo>
      </saml2:SubjectConfirmationData>
    </saml2:SubjectConfirmation>
  </saml2:Subject>
  <saml2:Conditions NotBefore="2012-02-23T07:28:50.178Z" NotOnOrAfter="2012-02-23T10:28:45.178Z">
    <saml2:AudienceRestriction>
      <saml2:Audience>http://wsp.someorg.dk</saml2:Audience>
    </saml2:AudienceRestriction>
  </saml2:Conditions>
  <saml2:AttributeStatement>
    <!–
```

```
      The attributes issued with identity token are added here. STS issues the attributes that are
specified in WSP's metadata (the WSP the identity token is requested for). Pls. refer section 2.4
"Attribute filtering" for more details.
      The attribute(s) below are listed as an example →
    <saml2:Attribute
      FriendlyName="CVR nummer"
      NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic"
      Name="dk:gov:saml:attribute:CvrNumberIdentifier">
      <saml2:AttributeValue xsi:type="xs:string">111111111</saml2:AttributeValue>
    </saml2:Attribute>
  </saml2:AttributeStatement>
</saml2:Assertion>
```

3.2 Local token case

The message below represents a request destined for STS issued by a WSC (Acme) using the local token case:

- The request includes a bootstrap token that was issued by WSC's local STS.
- The local STS have decided that the bootstrap token must be confirmed by a specific certificate (holder-of-key).
- The bootstrap token subject identifier is a MOCES certificate DN (the local STS know the relationship between corporate identities and MOCES certificates in this case).
- The bootstrap token contains the `AssuranceLevel` attribute, which is mandatory in the local token case. The bootstrap token also contains a `Mail` attribute. Both attributes will be copied to the identity token in accordance with Nemlog-in STS attribute filtering.
- The bootstrap token is signed by the local STS signing certificate.
- The request is signed by the holder-of-key certificate referenced in the bootstrap token `<saml2:SubjectConfirmationData>` element.
- The requested identity token is destined for the WSP: https://wsp.someorg.dk

```
<?xml version="1.0" encoding="utf-8"?>
<S11:Envelope>
  <S11:Header>
    <wsa:Action wsu:Id="action">http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/Issue</wsa:Action>
    <wsa:MessageID wsu:Id="msgid">uuid:aaaabbbb-cccc-dddd-eeee-aaaaaaaaaaaa</wsa:MessageID>
    <wsa:To wsu:Id="to">https://local.sts.nemlog-in.dk</wsa:To>
    <wsse:Security mustUnderstand="1">
      <wsu:Timestamp wsu:Id="sec-ts">
        <wsu:Created>2012-02-23T07:28:45Z</wsu:Created>
        <wsu:Expires>2012-02-23T08:28:45Z</wsu:Expires>
      </wsu:Timestamp>
      <wsse:BinarySecurityToken wsu:Id="sec-binsectoken"
        ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-
1.0#X509v3"
        EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-
1.0#Base64Binary">
```

```
        <!–
          X509 certificate used to sign the request →
      </wsse:BinarySecurityToken>
      <ds:Signature>
        <!–
          Signature over request –
          pls. refer request signature example for structure details →
      </ds:Signature>
    </wsse:Security>
  </S11:Header>
  <S11:Body wsu:Id=”body”>
    <wst:RequestSecurityToken Context=”urn:uuid:eb529fec-6498-11d7-b236-000629ba0001”>
      <wst:TokenType>
        http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0
      </wst:TokenType>
      <wst:RequestType>http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue</wst:RequestType>
      <wst14:ActAs>
        <!–
          The SAML assertion (bootstrap token) issued by the local STS →
        <saml2:Assertion ID=”_59ba7540-eaec-4b7a-a12e-d75073f3c001” IssueInstant=”2012-02-
23T07:28:45.578Z”
            Version=”2.0”>
          <saml2:Issuer>https://sts.acme.dk</saml2:Issuer>
          <ds:Signature>
            <!–
              Signature over assertion using local STS signing certificate
              pls. refer bootstrap token signature example for structure details →
          </ds:Signature>
          <saml2:Subject>
            <saml2:NameID Format=”urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName”>
              C=DK,O=ACME A/S // CVR:11111111,CN=Tola Kristiansen,Serial=CVR:11111111-RID:48245447
            </saml2:NameID>
            <saml2:SubjectConfirmation Method=”urn:oasis:names:tc:SAML:2.0:cm:holder-of-key”>
              <saml2:SubjectConfirmationData xsi:type=”saml2:KeyInfoConfirmationDataType”>
                <ds:KeyInfo>
                  <ds:X509Data>
                    <ds:X509Certificate>
                      <!–
                        X509 certificate that must sign the request to STS  →
                      </ds:X509Certificate>
                  </ds:X509Data>
                </ds:KeyInfo>
              </saml2:SubjectConfirmationData>
            </saml2:SubjectConfirmation>
          </saml2:Subject>
          <saml2:Conditions NotBefore=”2012-02-23T07:28:45.578Z” NotOnOrAfter=”2012-02-
23T08:28:45.578Z”>
            <saml2:AudienceRestriction>
              <saml2:Audience>http://local.sts.nemlog-in.dk</saml2:Audience>
            </saml2:AudienceRestriction>
          </saml2:Conditions>
          <saml2:AttributeStatement>
            <!–
                The attributes issued by the local STS are added here. Pls. refer section 2.4
“Attribute filtering” for more details.
              →
            <saml2:Attribute
              FriendlyName=”AssuranceLevel”
              NameFormat=”urn:oasis:names:tc:SAML:2.0:attrname-format:basic”
              Name=”dk:gov:saml:attribute:AssuranceLevel”>
              <saml2:AttributeValue xsi:type=”xs:string”>
                3
              </saml2:AttributeValue>
            </saml2:Attribute>
            <saml2:Attribute
              FriendlyName=”Mail”
              NameFormat=”urn:oasis:names:tc:SAML:2.0:attrname-format:basic”
              Name=”urn:oid:0.9.2342.19200300.100.1.3”>
```

```
            <saml2:AttributeValue xsi:type="xs:string">
              someinitials@acme.dk
            </saml2:AttributeValue>
          </saml2:Attribute>
        </saml2:AttributeStatement>
      </saml2:Assertion>
    </wst14:ActAs>
    <wsp:AppliesTo>
      <wsa:EndpointReference>
        <wsa:Address>https://wsp.someorg.dk</wsa:Address>
      </wsa:EndpointReference>
    </wsp:AppliesTo>
  </wst:RequestSecurityToken>
 </S11:Body>
</S11:Envelope>
```

The message below represents the response issued by STS in reply to the request above:

- The response includes an encrypted SAML assertion (identity token) issued by STS and destined for the WSP.
- The identity token is encrypted using WSP's X509 encrypting certificate and the token can only be decrypted with the corresponding private key.
- The identity token lifetime is defaulted to lifetime policy as the WSC didn't explicitly ask for a time range in the request.
- The response is signed by STS' signing certificate.

```
<?xml version="1.0" encoding="utf-8"?>
<S11:Envelope>
  <S11:Header>
    <wsa:Action wsu:Id="action">http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/Issue</wsa:Action>
    <wsa:MessageID wsu:Id="msgid">uuid:aaaabbbb-cccc-dddd-eeee-bbbbbbbbbbbb</wsa:MessageID>
    <wsa:RelatesTo wsu:Id="relto">uuid:aaaabbbb-cccc-dddd-eeee-aaaaaaaaaaaa</wsa:RelatesTo>
    <wsse:Security mustUnderstand="1">
      <wsu:Timestamp wsu:Id="sec-ts">
        <wsu:Created>2012-02-23T07:28:50Z</wsu:Created>
        <wsu:Expires>2012-02-23T15:28:50Z</wsu:Expires>
      </wsu:Timestamp>
      <ds:Signature>
        <!–
          Signature over response using Nemlog-in STS signing certificate –
          pls. refer response signature example for structure details →
      </ds:Signature>
    </wsse:Security>
  </S11:Header>
  <S11:Body wsu:Id="body">
    <wst:RequestSecurityTokenResponseCollection>
      <wst:RequestSecurityTokenResponse Context="urn:uuid:eb529fec-6498-11d7-b236-000629ba0001">
        <wst:TokenType>
          http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0
        </wst:TokenType>
        <wsp:AppliesTo>
          <wsa:EndpointReference>
            <wsa:Address>https://wsp.someorg.dk</wsa:Address>
          </wsa:EndpointReference>
        </wsp:AppliesTo>
        <wst:RequestedSecurityToken>
          <!–
```

```
          Encrypted SAML assertion are added here →
        <saml2:EncryptedAssertion>
          <xenc:EncryptedData wsu:Id="enc-token" Type="http://www.w3.org/2001/04/xmlenc#Element">
            <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc" />
            <ds:KeyInfo>
              <xenc:EncryptedKey>
                <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p">
                  <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
                </xenc:EncryptionMethod>
                <xenc:CipherData>
                  <xenc:CipherValue>
                    <!–
                      Encryption key encrypted using WSP's public key →
                  </xenc:CipherValue>
                </xenc:CipherData>
              </xenc:EncryptedKey>
            </ds:KeyInfo>
            <xenc:CipherData>
              <xenc:CipherValue>
                <!–
                  Encrypted SAML assertion
                  (The decrypted SAML assertion are documented in separate example) →
              </xenc:CipherValue>
            </xenc:CipherData>
          </xenc:EncryptedData>
        </saml2:EncryptedAssertion>
      </wst:RequestedSecurityToken>
      <wst:Lifetime>
        <wsu:Created>2012-02-23T07:28:50Z</wsu:Created>
        <wsu:Expires>2012-02-23T15:28:50Z</wsu:Expires>
      </wst:Lifetime>
    </wst:RequestSecurityTokenResponse>
  </wst:RequestSecurityTokenResponseCollection>
</S11:Body>
</S11:Envelope>
```

The message below represents the decrypted SAML assertion (identity token) included in the response message above:

- The assertion is signed by STS' signing certificate.
- The Subject identifier is exchanged to a persistent pseudonym by Nemlog-in STS as the WSP in this case accepts persistent pseudonyms.
- The certificate used for signing the request to STS is referenced as holder-of-key and must be used for signing the subsequent request to the WSP.
- The `AssuranceLevel` is copied from the bootstrap token. The `SpecVer` attribute is specified in WSP's metadata and therefore included (the `SpecVer` attribute is mandatory for the persistent pseudonym profile). The `Mail` attribute is not copied from the bootstrap token as the attribute is not allowed in the persistent pseudonym profile. Pls. refer section 2.4 "Attribute filtering" for more details on attribute handling.

```
<?xml version="1.0" encoding="utf-8"?>
<saml2:Assertion ID="_59ba7540-eaec-4b7a-a12e-d75073f3c002" IssueInstant="2012-02-23T07:28:50.178Z"
Version="2.0">
  <saml2:Issuer>
```

```
    https://local.sts.nemlog-in.dk
  </saml2:Issuer>
  <ds:Signature>
    <!–
      Signature over assertion using Nemlog-in STS signing certificate
      pls. refer assertion signature example for structure details →
  </ds:Signature>
  <saml2:Subject>
    <saml2:NameID Format=”urn:oasis:names:tc:SAML:2.0:nameid-format:persistent”>
      94gLThU5pC58Ki5ZcWUuw3e6lZIRLOpW9KL7bQgj/FU=
    </saml2:NameID>
    <saml2:SubjectConfirmation Method=”urn:oasis:names:tc:SAML:2.0:cm:holder-of-key”>
      <saml2:SubjectConfirmationData xsi:type=”saml2:KeyInfoConfirmationDataType”>
        <ds:KeyInfo>
          <ds:X509Data>
            <ds:X509Certificate>
              <!–
                X509 certificate that must sign subsequent request to WSP  →
            </ds:X509Certificate>
          </ds:X509Data>
        </ds:KeyInfo>
      </saml2:SubjectConfirmationData>
    </saml2:SubjectConfirmation>
  </saml2:Subject>
  <saml2:Conditions NotBefore=”2012-02-23T07:28:50.178Z” NotOnOrAfter=”2012-02-23T15:28:50.178Z”>
    <saml2:AudienceRestriction>
      <saml2:Audience>http://wsp.someorg.dk</saml2:Audience>
    </saml2:AudienceRestriction>
  </saml2:Conditions>
  <saml2:AttributeStatement>
    <!–
      The attributes issued with identity token are added here. STS issues the attributes that are
specified in WSP’s metadata (the WSP the identity token is requested for). Attributes from the
bootstrap token are copied to the identity token by STS – as WSP in this case uses the persistent
NameID format a limited set of attributes are supported. Pls. refer section 2.4 “Attribute filtering”
for more details. →
    <saml2:Attribute
      FriendlyName=”AssuranceLevel”
      NameFormat=”urn:oasis:names:tc:SAML:2.0:attrname-format:basic”
      Name=”dk:gov:saml:attribute:AssuranceLevel”>
      <saml2:AttributeValue xsi:type=”xs:string”>
        3
      </saml2:AttributeValue>
    </saml2:Attribute>
    <saml2:Attribute
      FriendlyName=”SpecVer”
      NameFormat=”urn:oasis:names:tc:SAML:2.0:attrname-format:basic”
      Name=”dk:gov:saml:attribute:SpecVer”>
      <saml2:AttributeValue xsi:type=”xs:string”>
        2.0
      </saml2:AttributeValue>
    </saml2:Attribute>
  </saml2:AttributeStatement>
</saml2:Assertion>
```

3.3 Signature case

The message below represents a request destined for STS issued by an application in some organization using the signature case:

- The request does not include a bootstrap token.
- The request is signed by the user with his/her MOCES certificate.
- The requested identity token is destined for the WSP: https://wsp.someorg.dk

```
<?xml version="1.0" encoding="utf-8"?>
<S11:Envelope>
  <S11:Header>
    <wsa:Action wsu:Id="action">http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/Issue</wsa:Action>
    <wsa:MessageID wsu:Id="msgid">uuid:aaaabbbb-cccc-dddd-eeee-aaaaaaaaaaaa</wsa:MessageID>
    <wsa:To wsu:Id="to">https://´signature.sts.nemlog-in.dk</wsa:To>
    <wsse:Security mustUnderstand="1">
      <wsu:Timestamp wsu:Id="sec-ts">
        <wsu:Created>2012-02-23T07:28:45Z</wsu:Created>
        <wsu:Expires>2012-02-23T08:28:45Z</wsu:Expires>
      </wsu:Timestamp>
      <wsse:BinarySecurityToken wsu:Id="sec-binsectoken"
        ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-
1.0#X509v3"
        EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-
1.0#Base64Binary">
        <!–
        MOCES X509 certificate used to sign the request to STS →
      </wsse:BinarySecurityToken>
      <ds:Signature>
        <!–
        Signature over request –
        pls. refer request signature example for structure details →
      </ds:Signature>
    </wsse:Security>
  </S11:Header>
  <S11:Body wsu:Id="body">
    <wst:RequestSecurityToken Context="urn:uuid:eb529fec-6498-11d7-b236-000629ba0001">
      <wst:TokenType>
        http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0
      </wst:TokenType>
      <wst:RequestType>http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue</wst:RequestType>
      <wsp:AppliesTo>
        <wsa:EndpointReference>
          <wsa:Address>https://wsp.someorg.dk</wsa:Address>
        </wsa:EndpointReference>
      </wsp:AppliesTo>
      <wst:Lifetime>
        <wsu:Expires>2012-02-23T10:28:45Z</wsu:Expires>
      </wst:Lifetime>
    </wst:RequestSecurityToken>
  </S11:Body>
</S11:Envelope>
```

The message below represents the response issued by STS in reply to the request above:

- The response includes an encrypted SAML assertion (identity token) issued by STS and destined for the WSP.
- The identity token is encrypted using WSP's X509 encrypting certificate and the token can only be decrypted with the corresponding private key.
- The response is signed by STS' signing certificate.

```
<?xml version="1.0" encoding="utf-8"?>
<S11:Envelope>
  <S11:Header>
    <wsa:Action wsu:Id="action">http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/Issue</wsa:Action>
    <wsa:MessageID wsu:Id="msgid">uuid:aaaabbbb-cccc-dddd-eeee-bbbbbbbbbbbb</wsa:MessageID>
    <wsa:RelatesTo wsu:Id="relto">uuid:aaaabbbb-cccc-dddd-eeee-aaaaaaaaaaaa</wsa:RelatesTo>
    <wsse:Security mustUnderstand="1">
      <wsu:Timestamp wsu:Id="sec-ts">
        <wsu:Created>2012-02-23T07:28:50Z</wsu:Created>
        <wsu:Expires>2012-02-23T10:28:45Z</wsu:Expires>
      </wsu:Timestamp>
      <ds:Signature>
        <!–
          Signature over response using STS signing certificate –
          pls. refer response signature example for structure details →
      </ds:Signature>
    </wsse:Security>
  </S11:Header>
  <S11:Body wsu:Id="body">
    <wst:RequestSecurityTokenResponseCollection>
      <wst:RequestSecurityTokenResponse Context="urn:uuid:eb529fec-6498-11d7-b236-000629ba0001">
        <wst:TokenType>
          http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0
        </wst:TokenType>
        <wsp:AppliesTo>
          <wsa:EndpointReference>
            <wsa:Address>https://wsp.someorg.dk</wsa:Address>
          </wsa:EndpointReference>
        </wsp:AppliesTo>
        <wst:RequestedSecurityToken>
          <!–
            Encrypted SAML assertion are added here →
          <saml2:EncryptedAssertion>
            <xenc:EncryptedData wsu:Id="enc-token" Type="http://www.w3.org/2001/04/xmlenc#Element">
              <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc" />
              <ds:KeyInfo>
                <xenc:EncryptedKey>
                  <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p">
                    <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
                  </xenc:EncryptionMethod>
                  <xenc:CipherData>
                    <xenc:CipherValue>
                      <!–
                        Encryption key encrypted using WSP's public key →
                    </xenc:CipherValue>
                  </xenc:CipherData>
                </xenc:EncryptedKey>
              </ds:KeyInfo>
              <xenc:CipherData>
                <xenc:CipherValue>
                  <!–
                    Encrypted SAML assertion
                    (The decrypted SAML assertion are documented in separate example) →
                </xenc:CipherValue>
              </xenc:CipherData>
            </xenc:EncryptedData>
          </saml2:EncryptedAssertion>
        </wst:RequestedSecurityToken>
        <wst:Lifetime>
          <wsu:Created>2012-02-23T07:28:50Z</wsu:Created>
          <wsu:Expires>2012-02-23T10:28:45Z</wsu:Expires>
        </wst:Lifetime>
      </wst:RequestSecurityTokenResponse>
    </wst:RequestSecurityTokenResponseCollection>
  </S11:Body>
</S11:Envelope>
```

The message below represents the decrypted SAML assertion (identity token) included in the response message above:

- The assertion is signed by STS' signing certificate.
- The Subject identifier is the MOCES certificate DN from the certificate that signed the request to STS. The WSP accepts in this case X509SubjectName.
- The CVR-number attribute is the only attribute specified in WSP's metadata, hence only this attribute is included.

```
<?xml version="1.0" encoding="utf-8"?>
<saml2:Assertion ID="_59ba7540-eaec-4b7a-a12e-d75073f3c002" IssueInstant="2012-02-23T07:28:50.178Z"
Version="2.0">
  <saml2:Issuer>
    https://signature.sts.nemlog-in.dk
  </saml2:Issuer>
  <ds:Signature>
    <!—Signature over assertion using STS signing certificate
    pls. refer assertion signature example for structure details →
  </ds:Signature>
  <saml2:Subject>
    <saml2:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName">
      C=DK,O=ACME A/S // CVR:11111111,CN=Tola Kristiansen,Serial=CVR:11111111-RID:48245447
    </saml2:NameID>
    <saml2:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
      <saml2:SubjectConfirmationData />
    </saml2:SubjectConfirmation>
  </saml2:Subject>
  <saml2:Conditions NotBefore="2012-02-23T07:28:50.178Z" NotOnOrAfter="2012-02-23T10:28:45.178Z">
    <saml2:AudienceRestriction>
      <saml2:Audience>http://wsp.someorg.dk</saml2:Audience>
    </saml2:AudienceRestriction>
  </saml2:Conditions>
  <saml2:AttributeStatement>
    <!—
      The attributes issued with identity token are added here. STS issues the attributes that are
specified in WSP's metadata (the WSP the identity token is requested for). Pls. refer section 2.4
"Attribute filtering" for more details.
      The attribute(s) below are listed as an example →
    <saml2:Attribute
      FriendlyName="CVR nummer"
      NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic"
      Name="dk:gov:saml:attribute:CvrNumberIdentifier">
      <saml2:AttributeValue xsi:type="xs:string">111111111</saml2:AttributeValue>
    </saml2:Attribute>
  </saml2:AttributeStatement>
</saml2:Assertion>
```

3.4 Request signature

The message below represents the signature element that is included in requests to Nemlog-in STS.

- The signature covers all request elements including the certificate used for signing request, which is included as a binary security token.

```
<ds:Signature>
  <ds:SignedInfo>
    <ds:CanonicalizationMethod Algorithm=”http://www.w3.org/2001/10/xml-exc-c14n#”/>
    <ds:SignatureMethod Algorithm=”http://www.w3.org/2000/09/xmldsig#rsa-sha1”/>
    <!–
    Signature must cover all message elements →
    <ds:Reference URI=”#action”>
      <ds:Transforms>
        <ds:Transform Algorithm=”http://www.w3.org/2000/09/xmldsig#envelopedsignature”/>
      </ds:Transforms>
      <ds:DigestMethod Algorithm=”http://www.w3.org/2000/09/xmldsig#sha1”/>
      <ds:DigestValue>
        <!–digest value →
      </ds:DigestValue>
    </ds:Reference>
    <!–
    The remaining elements are handled like the #action reference
    above→
    <ds:Reference URI=”#msgid”>…</ds:Reference>
    <ds:Reference URI=”#to”>…</ds:Reference>
    <ds:Reference URI=”#sec-ts”>…</ds:Reference>
    <ds:Reference URI=”#sec-binsectoken”>…</ds:Reference>
    <ds:Reference URI=”#body”>…</ds:Reference>
  </ds:SignedInfo>
  <ds:SignatureValue>
    <!–Signature value →
  </ds:SignatureValue>
  <ds:KeyInfo>
    <wsse:SecurityTokenReference>
      <wsse:Reference URI=”#sec-binsectoken”/>
    </wsse:SecurityTokenReference>
  </ds:KeyInfo>
</ds:Signature>
```

3.5 Response signature

The message below represents the signature element that is included in responses issued from Nemlog-in STS.

- The signature covers all response elements.
- The signature is always given with the Nemlog-in STS signing certificate.

```
<ds:Signature>
  <ds:SignedInfo>
    <ds:CanonicalizationMethod Algorithm=”http://www.w3.org/2001/10/xml-exc-c14n#”/>
    <ds:SignatureMethod Algorithm=”http://www.w3.org/2000/09/xmldsig#rsa-sha1”/>
    <!–
    Signature must cover all message elements →
    <ds:Reference URI=”#action”>
      <ds:Transforms>
        <ds:Transform Algorithm=”http://www.w3.org/2000/09/xmldsig#envelopedsignature”/>
      </ds:Transforms>
      <ds:DigestMethod Algorithm=”http://www.w3.org/2000/09/xmldsig#sha1”/>
```

```
      <ds:DigestValue>
        <!–digest value →
      </ds:DigestValue>
    </ds:Reference>
    <!–
    The remaining elements are handled like the #action reference
    above→
    <ds:Reference URI=”#msgid”>…</ds:Reference>
    <ds:Reference URI=”#relto”>…</ds:Reference>
    <ds:Reference URI=”#sec-ts”>…</ds:Reference>
    <ds:Reference URI=”#body”>…</ds:Reference>
  </ds:SignedInfo>
  <ds:SignatureValue>
    <!–Signature value →
  </ds:SignatureValue>
</ds:Signature>
```

3.6 Assertion signature

The message below represents the signature element that is included in assertions (bootstrap tokens and identity tokens) included with requests and responses between parties.

- The signature covers the assertion element.

```
<ds:Signature>
  <ds:SignedInfo>
    <ds:CanonicalizationMethod Algorithm=”http://www.w3.org/2001/10/xml-exc-c14n#”/>
    <ds:SignatureMethod Algorithm=”http://www.w3.org/2000/09/xmldsig#rsa-sha1”/>
    <!–
    Signature must cover assertion element →
    <ds:Reference URI=”<!–saml2:Assertion ID attribute→”>
      <ds:Transforms>
        <ds:Transform Algorithm=”http://www.w3.org/2000/09/xmldsig#envelopedsignature”/>
      </ds:Transforms>
      <ds:DigestMethod Algorithm=”http://www.w3.org/2000/09/xmldsig#sha1”/>
      <ds:DigestValue>
        <!–digest value →
      </ds:DigestValue>
    </ds:Reference>
    <ds:SignatureValue>
      <!–Signature value →
    </ds:SignatureValue>
  </ds:Signature>
```

4 References

The following references are used throughout the document:

Reference	Description
WST	OASIS WS-Trust 1.3: http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-os.html
XMLSIG	W3C XML Signature: http://www.w3.org/TR/xmldsig-core/
OIO-WST	OIO WS-Trust Profile 1.0.1: http://digitaliser.dk/resource/516724
OIO-WST-DEPL	OIO WS-Trust Deployment Profile 1.0: http://digitaliser.dk/resource/516724
BILAG3	KFOBS etableringskontrakt, bilag 3 leverancebeskrivelse: https://bpp.nnit.com/sites/DolphinDK/kfobs/Kontrakt/Etableringskontrakt/Bilag%203%20Leverancebeskrivelse.doc
OIO-BOOT-TOKEN	OIO Bootstrap Token Profile 1.0.1: http://digitaliser.dk/resource/516724
OIO-IDENT-TOKEN	OIO SAML Profile for Identity Tokens 1.0: http://digitaliser.dk/resource/516724
OIO-WEB-SSO	OIO Web SSO Profile 2.0.9: http://digitaliser.dk/resource/2377872
OIO-BASIC-PRIV	OIO SAML Basic Privilege Profile 1.0.1: http://digitaliser.dk/resource/2377872
Xs:ID	The `xs:ID` simple type is used to declare SAML identifiers for assertions, requests, and responses. Values declared to be of type `xs:ID` in this specification must satisfy the following properties in addition to those imposed by the definition of the `xs:ID` type itself: • Any party that assigns an identifier must ensure that there is negligible probability that that party or any other party will accidentally assign the same identifier to a different data object. • Where a data object declares that it has a particular identifier, there must be exactly one such declaration.

Reference	Description
	All SAML time values have the type `xs:dateTime`, which is built in to the W3C XML Schema Datatypes specification, and MUST be expressed in UTC form, with no time zone component. SAML system entities should not rely on time resolution finer than milliseconds. Implementations´must not generate time instants that specify leap seconds.
XMLENC	W3C XML Encryption Syntax and Processing: http://www.w3.org/TR/xmlenc-core/

4.1 Namespace references

The following namespace references are used throughout the document:

Reference	Namespace
ds	http://www.w3.org/2000/09/xmldsig#
S11	http://www.w3.org/2003/05/soap-envelope
wsa	http://schemas.xmlsoap.org/ws/2004/03/addressing
wsp	http://schemas.xmlsoap.org/ws/2002/12/policy
wsse	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd
wst	http://docs.oasis-open.org/ws-sx/ws-trust/200512
wst14	http://docs.oasis-open.org/ws-sx/ws-trust/200802
wsu	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd
saml2	urn:oasis:names:tc:SAML:2.0:assertion
xenc	http://www.w3.org/2001/04/xmlenc#
xs	http://www.w3.org/2001/XMLSchema

5 Change log

Date	Version	Description of Changes	Initials
2012-09-08	0.a	Document Created	MWL
2012-10-11	0.b	Added rules for "Request processing rules" section (section is not complete yet).	MWL
2013-02-08	0.c	Updated the "Request processing" section.	MWL
2013-02-22	0.d	Document updated according to workshop #1 findings.	MWL/ TmLN
2013-02-28	0.e	Document updated according to workshop #2 findings.	MWL
2013-03-12	0.f	Added response message section and sections related to this. The document is ready for workshop #3.	MWL
2013-03-13	0.g	Document updated according to workshop #3 findings. Added normative examples. Document ready for final review	MWL
2013-04-15	0.f	Updated according to final review.	MWL
2013-04-16	1.0	Approved by DIGST (Henrik Robert Langer mail dated 2013-04-16)	MWL
2013-05-21	1.a	urn:oasis:names:tc:SAML:2.0:nameid-format:X509SubjectName changed to urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName in section "2.3.2.4.6.1 /saml2:Assertion/saml2:Subject/saml2:NameID" Removed attribute urn:oid:2.5.4.65 (OCES Pseudonym) from list of supported attributes in accordance with change request GENEREL.10, which has been approved by DIGST.	TMLN, MWL
2013-09-18	1.b	New attribute filtering rule added to section "Processing rule for local token case" Updated request validation section and normative examples	TMLN, AXPE

Date	Version	Description of Changes	Initials
2014-02-26	1.c	Corrected casing of dk:gov:saml:attribute:SENumberIdentifier ref. OIO Web SSO Profile v2.09	AxPe
2014-03-17	1.d	Updated documents according to comments (local token attribute casing and bootstrap base64 encoding)	AxPe
2014-04-03	2.0	Approved by DIGST	AxPe