Signature Profile for BankID

Version: 2.3

2016-02-18

Table of Content

1 Introduction

1.1 Revisions

Date	Version	Description	Author
2008-01-14 - - 2012-06-20	1.0 – 1.6	Historical versions.	BankID
2014-01-19	2.0	Changed to comply with the new BankID solution where relying parties access the BankID RP Interface. Signatures created using the old pluginbased solution is described in previous versions of this document. Changes: References to BICS removed. Some elements in bankIDSignedData removed. Others added. Removed chapter “Mobile Signatures”. There is no need to highlight Mobile Signatures.Made a note related to the Id-attribute in BankID SignedData Minor editorial.	Jesper Skagerberg
2014-06-10	2.1	Notes related to signatures created using a class 4 reader.	Jesper Skagerberg
2015-05-29	2.2	usrVisibleData is always UTF8 clientInfo.rpRef is a new element for the PC client. The functionality to sign files is deprecated. Minor editorial. Changes related to new mobile clients which creates signatures with the same content as the PC client: • srvInfo.displayName • clientInfo.env.ai.utb • clientInfo.env.ai.fsib • clientInfo.env.ai.requirement	Jesper Skagerberg
2016-02-18	2.3	srvInfo.name: the content is a combination RFC4514 and RFC4519 srvInfo.displayName: is not included in old Mobile Clients A new element clientInfo.env.ai.uauth was introduced Editorial	Jesper Skagerberg

1.2 References

ID	Reference
[XML-SIG]	XML-Signature Syntax and Processing, W3C Recommendation 12 February 2002

2 General Description

The signature created by the BankID solution is an XML Signature according to [XML-SIG]. To make it possible to implement functionality to verify the signatures without having to implement the complete [XML-SIG] including all references, we have limited the scope of [XML-SIG]. This limited scope is defined in this document and is referred to as the “Signature Profile for BankID”.

We foresee two typical use cases:

- Perform authentication and digitally sign transactions. This is the typical BankID use case.
- Let the user sign files and documents created by the server (this functionality is deprecated and should not be used).

2.1 Signature Profile

- The ***Signature*** element is always enveloping an ***Object***.
- The ***Object*** element in turn always contains a ***bankIdSignedData*** element with the ***Id*** attribute set to *bidSignedData*, specified below.
- **One and only one** reference to ***bankIDSignedData*** is always included in the ***SignedInfo*** element. As reference, the Id *bidSignedData* is used. Transformation is according to: http://www.w3.org/TR/2*001/REC-xml-c*14n-20010315. The reference always includes the attribute *Type* (to indicate special processing of the signed information).
- One and only *one* reference to ***KeyInfo*** is always included in the ***SignedInfo*** element. As reference, the Id *bidKeyInfo* is used. This is to connect the certificates with the signature. Transformation is according to: http://www.w3.org/TR/2001/REC-xml-c14n-20010315.
- If a file is signed, one and only one reference to an external resource (file) is included in the ***SignedInfo*** element. The reference always starts with file:/// to indicate that the reference is not general available from any host. The name of the resource (the file) will be base64-encoded. Note that the base64 encoding may result in characters that are not formally allowed in a file URI.
- Canonicalization is according to: http://www.w3.org/TR/2001/REC-xml-c14n-20010315.
- Message digesting is: http://www.w3.org/2001/04/xmlenc#sha256.
- Signature algorithm is: http://www.w3.org/2000/09/xmldsig#rsa-sha1 **or** http://www.w3.org/2001/04/xmldsig-more#rsa-sha256.
- The ***KeyInfo*** element holds the signers certificate together with ***the certificat***e chain (excluding the trusted root). The certificates is sorted (when read from top, the user certificate comes first followed by the intermediate ***CA-***certificates in order).

2.2 Overview Example

```
<Signature>
	<SignedInfo>
		<CanonicalizationMethod></CanonicalizationMethod>
		<SignatureMethod></SignatureMethod>
		<Reference Type="http://www.bankid.com/signature/v1.0.0/types" URI="#bidSignedData">
			<Transforms></Transforms>
			<DigestMethod></DigestMethod>
			<DigestValue></DigestValue>
		</Reference>
		<Reference URI="file:///YXZ0YWwteHl6MTIzLnBkZg==">
			<DigestMethod></DigestMethod>
			<DigestValue></DigestValue>
		</Reference>
		<Reference URI="#bidKeyInfo">
			<Transforms></Transforms>
			<DigestMethod></DigestMethod>
			<DigestValue></DigestValue>
		</Reference>
	</SignedInfo>
	<SignatureValue></SignatureValue>
	<KeyInfo Id=”bidKeyInfo”></KeyInfo>
	<Object>
		<bankIdSignedData Id="bidSignedData">
		</bankIdSignedData>
	</Object>
</Signature>
```

2.3 Overview description of the Signature element

The ***SignedInfo*** element specifies what was signed and with what algorithms.

The ***SignatureMethod*** and ***CanonicalizationMethod*** elements specifies how the ***SignatureValue*** element has been computed.

A list of ***Reference*** elements specify which resources that have been digested. These elements also specifies any transforms to apply to the resource before applying the hash, the digest (hash) algorithm (in ***DigestMethod***), and the result of applying it to the resource (Base64-encoded in ***DigestValue***).

The ***SignatureValue*** element holds the Base64-encoded value of the signature. This value is the signature (produced according to the ***SignatureMethod*** element) of the ***SignedInfo*** element after serializing it with the algorithm specified by the ***CanonicalizationMethod*** element.

The ***KeyInfo*** element contains a set of X.509 certificates.

The ***bankIdSignedData*** element contains the BankID specific data that is secured using a digital signature, and is referred to from one of the references. As reference, the *Id* attribute set to *bidSignedData* is used. The content of this element is specified below. The main elements in ***bankIdSignedData*** are:

- ***usrVisibleData*** - Data that is displayed to the user (by the client software or card reader) at time of signature computation. This element represents the typical online use case for BankID signed transaction in which the user sees what he or she signs.
- ***usrNonVisibleData*** – Data that is not displayed to the user at time of signature computation. Introducing this element definitely violates the "What You See is What You Sign" and must be used with care. However, in some use cases it is beneficial if the data is not explicitly presented to the user. Introducing "visible" and "non visible" data will make it very clear what the user was able to see and not.
- ***srvInfo*** – Additional information generated by the server, e.g. challenge or nonce.
- ***clientInfo*** – Information generated by the client at time of signature computation, e.g. client software versions.

2.4 Content of bankIDSignedData

Element	Description	Type	Rules for Authentication	Rules for Signature
UsrVisibleData				
usrVisibleData	The content of this element represents information that was presented to the user at time of signing. **Attribute**. The attribute "charset" specifies the character set used for the information. Charset is "UTF-8". **Attribute**. The attribute "visible" is used as a holder of a specific string. This specific string may be included in order to provide support for certain card readers with multi line display capabilities.	Base64 encoded	Prohibited	Mandatory
UsrNonVisibleData				
usrNonVisibleData	The content of this element represents information that was **not** presented to the user at time of signing.	Base64 encoded	Prohibited	Optional
srvInfo	**Additional elements may be added in future releases**			
srvInfo	This element holds information generated by the server. The information is sent to the client software together with the data to be signed.	Complex	Mandatory	Mandatory
srvInfo.nonce	The element holds the challenge (if the signature is a result of an authentication operation) or the nonce (if the signature is a result of a signature operation).	Base64 encoded	Mandatory	Mandatory
srvInfo.name	The name of the relying party requesting login or signing. The type is "Base64 encoded". Holds the base64 encoded value of the string representation of distinguished name, according to RFC4514 with attribute names for 2.5.4.41 ("name") and 2.5.4.5 ("serialNumber") from RFC4519, of the subject in the relying party certificate. A subset of the name (the "friendly name") is displayed to the user.	Base64 encoded	Mandatory	Mandatory
srvInfo.displayName	For the desktop client and new versions of the mobile client[1]. The name of the relying party requesting login or signing. This name is displayed to the user.	Base64 encoded	Optional	Optional
clientInfo	**Additional elements may be added in future releases**			
clientInfo	Holds information generated by the client software at the time of signature computation.	Complex	Mandatory	Mandatory

[1] Beginning may 2015 new versions of the mobile client was launched. Signatures created using these versions will have the same content as signatures created using the desktop client. During a transition period both types will occur.

clientInfo.funcId	The functional identifier identifies the purpose of the signing operation.	String	Mandatory set to "Identification"	Mandatory set to "Signing"
clientInfo.version	The version of the client software. For signatures created using mobile devices this is the version number of the client software. For desktop clients this is the versionstring.	Base64 encoded	Mandatory	Mandatory
clientInfo.rpRef	Only for the desktop client. The reference used when the client software was automatically started with the bankid scheme. The typical usecase is to use a hashsum of a file as reference. Min 8 bytes. Max 256 bytes (encoded).	Bas64 encoded	Optional	Optional
clientInfo.env	Describes the environment in which the client software executes.	Complex	Mandatory	Mandatory
clientInfo.env.ai	Assessment Information. Holds information, gathered by the client about the environment, that can be used by the relying party to make assessments about the signature.	Complex	Mandatory	Mandatory
clientInfo.env.ai.uhi	Unique Hardware ID. An ID, generated by the client software, unique for the hardware on which the client software is installed.	Base64 encoded	Mandatory	Mandatory
clientInfo.env.ai.fsib	For the desktop client and new versions of the mobile client[2]. Indicates if the "secure desktop" feature was active in the Client Software when the transaction was signed. The content is "0" (secure desktop was not active) or "1" (secure desktop was active). If the element is not present the secure desktop feature was active.	Enumeration	Optional	Optional
clientInfo.env.ai.utb	For the desktop client and new versions of the mobile client[1]. Used Token Bearer. Information about the bearer for the used token. This information can be one (and only one) of the strings "file" (for file based tokens), "cr1", "cr2", "cr3", "cr4" (for card based tokens) and "cs1" (for file based tokens and new versions of mobile clients) or "unknown".	Enumeration	Optional	Optional
clientInfo.env.ai.type	Holds the used device type (in example ANDROID, IPHONE or WINDOWS).	Base64 encoded	Mandatory	Mandatory
clientInfo.env.ai.deviceinfo	Depends on device e.g. "win7" or "Samsung Corp, Galaxy, 2.3". If no information is available then this is set to "unknown".	Base64 encoded	Mandatory	Mandatory
clientInfo.env.ai.policy	Only for old versions of the mobile client[1]. Indicates which policy the transaction was created according to. If no special policy is used the element is omitted. If OTP is used the element is included and has the value "1.2.752.78.5.2".	String	Optional	Optional

[2] Beginning may 2015 new versions of the mobile client was launched. Signatures created using these versions will have the same content as signatures created using the desktop client. During a transition period both types will occur.

clientInfo.env.ai.requirement	For the desktop client and new versions of the mobile client[1]. Indicates which requirement the transaction was created according to. The element holds one or more conditions of different types and values. Example 1: A signature created using a Mobile BankID: <condition> <type>CertificatePolicies</type> <value>1.2.752.78.1.5</value> </condition> Example 2: A signature created using a BankID on a smart card in a class 2 card reader: <condition> <type>CertificatePolicies</type> <value>1.2.752.78.1.2</value> </condition> <condition> <type>CardReader</type> <value>class2</value> </condition>	Complex	Optional	Optional
clientInfo.env.ai.requirement.condition	A condition that was fulfilled when the signature was created.	Complex	Mandatory	Mandatory
clientInfo.env.ai.requirement.condition.type	The type of condition	String	Mandatory	Mandatory
clientInfo.env.ai.requirement.condition.value	The value of the condition	String	Mandatory	Mandatory
clientInfo.env.ai.uauth	For new versions of the mobile client. Indicates how the user was authenticated. Possible values are “fp1” (fingerprint sensor type 1) and “pw” (password/pin). If this element is not present password/pin was used.	Enumeration	Optional	Optional

3 Class 4 readers

This section is included to describe how use of class 4 readers affects the signature profile. Only Vasco DP920 is supported.

3.1 Changes in Content of the Signature element

- The ***Object*** element also contains a ***bankIdUnsignedData*** element.
- One and only one reference to the element ***usrVisibleData*** **must** be included in the ***SignedInfo*** element. This reference is always included last in the list of references. As reference, the Id *swys* is used. Due to an error in the Vasco DP920 reader this reference is not possible to verify using standard methods. The transformation is not correct. To verify the refererence you need to compute the digest over the base64-decoded (to win-1252) value (which is base64 encoded in the signature).
- The ***SignatureValue*** element has an Id attribute set to *bidSignatureValue*. This is used to be able to refer to it from the signature created by the card reader.

3.2 Changes in content in bankIDSignedData element

The element usrVisibleData will have an additional attribute "Id" with value *swys* used to make it referable.

The element clientInfo will have an additional complex type utbInfo needed to verify the counter signature created by the cardreader.

3.3 New element, bankIdUnsignedData

The ***bankIdUnsigedData*** element is present only when the signature is created using a class 4 card reader. When present, the element contains a counter signature, created by the card reader, encompassing the value in the *SignatureValue* element and some additional information created by the reader.

The ***bankIdUnsignedData*** element is not referenced to from ***SignedInfo***.

The content of ***bankIdUnsignedData*** is not further described in this document.

3.4 Card readers classes

We use the following definition of classes of card readers.

- Class 1. A.k.a. "transparent card readers". Significant for these card readers is that the visible text to be signed "usrVisibleData" is displayed in the client software, not by the card reader, and that the PIN code is entered via the client software, not through a PIN pad on the card reader.
- Class 2. Card readers with a built in PIN pad. Significant is that the visible text to be signed "usrVisibleData" is displayed in the client software, not by the card reader, but the PIN code is entered through a PIN pad on the card reader, not via the client software.
- Class 3. Card readers with built in PIN pad and extended display. Functionality in the reader guarantees that the information that is displayed also is included in the signature. Sometimes called "SWYS" card reader (See What You Sign/Sign What You See).
- Class 4. In addition to the same properties and functionality as a class 3 reader, a class 4 reader also provide its own signature. The class 4 card reader signature encompass the (value of the) main signature.

Note that a specific card reader may be compatible with more than one class and may be able to switch between different classes depending on how the client software communicates with it.

4 Verification

Verification of the signatures is done by the BankID Server. Relying parties that wants to verify the signatures themselves may have interest in the following information.

- Signatures must be verified according to “3.2 Core Validation” in [XML-SIG].
- The <signedInfo> is in canonical form. Applying the defined canonalisation to it does not change content.
- In the same way, the content of the references (<keyInfo> and <bankIdSignedData>) are transformed. Applying the defined transformation to them does not change content.

This implies that it should be possible to compute the digest of the references without transforming them, and to verify <signedInfo> without applying any canonalisation. However, the canonalisation and transform identifiers are included to be precise in the specification, and the verifying software must verify them being correct.

4.1 The Id="bidSignedData"

Some versions of XML-parsers may have difficulties parsing the ”Id” attribute in “bankIdSignedData” since it exists in the BankID namespace, but is not defined in the BankId namespace. document.getElementById may not return correct information and core validation will fail. See http://santuario.apache.org/java150releasenotes.html, "Major changes to how Elements are resolved".

One workaround is to use:

domValidateContext.setIdAttributeNs(element, null, "Id");