Danske Bank

Encryption, Signing and Compression in Financial Web Services

Details of how to call the Danske Bank financial web service

Table of Contents

Version history

Version	Date	Description of change	Author
2.2	5. January 2010	RFC 1952 added to clarify gzip. Section on future changes removed. Example XML added. Section on signatures added.	Mikkel T. Jensen
2.3	22. February 2010	RSA-OAEP replaced by RSA-v1.5. More requirements to signatures added.	Mikkel T. Jensen
2.4	16. April 2010	Section added on future changes to the financial WS specification. Introduction changed.	Mikkel T. Jensen
2.4.1	4. March 2010	Example XML corrected to use RSA 1.5.	Mikkel T. Jensen
2.4.2	18. April 2012	Corrected Introduction text	Christian Enevold
2.4.3	29. September 2012	Changes due to renaming of Sampo Pankki to Danske Bank.	Mikkel T. Jensen
2.4.4	4. December 2012	Changes to URL.	Lea Troels Møller Pedersen
2.4.5	20. August 2014	Corrected mistake with optional encryption	Mikkel T. Jensen
2.4.6	16. September 2014	Corrected diagram	Mikkel T. Jensen
2.4.7	12. May 2015	Error removed from transport signature example XML.	Mikkel T. Jensen
2.4.8	27. July 2017	Added references to EDI WS and PKI documents	Andreja Andric
2.5.0	15. February 2023	Added an introductory recap of the integration process. Fixed broken bookmarks and cross-references. Removed references of an old version of the Financial Messages specification document.	Andreja Andric
2.6.0	5. September 2023	Updated algorithms.	Andreja Andric

Introduction

The Danske Bank Web Services solution is build on the Web Services specifications from the Federation of Finnish Financial services [1].

This document clarifies the use of XML encryption and compression and how it is implemented in Danske Bank.

Principal aim of this document is to cover the details of the security elements in the first group (EDI Web Services [2]). However, the requirements on XML encryption and enveloped signatures also apply to the second group (PKI Service [3]).

Overview of the Integration Process

As we have seen in the introductory page on our website (danskeci), the solution is divided into two categories of service:

1) EDI Web Services

2) PKI Web Services

The first regards sending and receiving authenticated and encrypted files. The second regards certificate management, whose aim is to maintain the communication secure. Because the security is indispensable for communication, you have to start your integration process with PKI Web Services.

First you have to obtain the bank certificate. Therefore, the first call you have to implement is GetBankCertificate. This message you neither sign nor encrypt, as the bank certificate is available publicly. You will use this certificate to encrypt your further messages to the bank. Moreover, we recommend that, in further communication with the bank, you make this call on a regular basis, in order to make sure that you always have the most recent version of the certificate. To make sure the bank certificate is correct, you can verify it using the Danske Bank Root certificate, provided on our danskeci website.

Next, you have to create your own certificate, by calling CreateCertificate. Your message will contain a secret four digit pin, and for this reason you'll have to encrypt the message using the public bank certificate you obtained in the previous call.

When you have created your own certificate, you are ready to start making calls to EDI Web Services. However we recommend to implement at least RenewCertificate call as well, as you will need it when your current certificate expires.

Examples of all the messages are given on our danskeci website. Some of the messages are also reproduced in the Appendices to this document, for easier reference. However, avoid copying and pasting the examples from this document, because some characters, like dash ('-') might be interpreted incorrectly when pasting. Use the examples from our danskeci website instead.

Regarding encryption

Our implementation of encryption

XML encryption is applied at the ApplicationRequest/ApplicationResponse level. This means that the ApplicationRequest (or ApplicationResponse) element is replaced by an EncryptedData element from the XML encryption standard. The EncryptedData contains the actual encrypted data (in a CipherData subelement) as well as sub-elements necessary for decryption (EncryptionMethod and KeyInfo). The reason this approach can work is that the ApplicationRequest/ApplicationResponse/EncryptedData element is base64 encoded when the actual web service call is made. This means that there is no schema-check of the ApplicationRequest/Response schema on the web service level. The schema-check of the ApplicationRequest/Response should not be done until the actual XML has been reconstructed using base64 decoding, followed by decryption. The order of processing when receiving a web service call should be as follows:

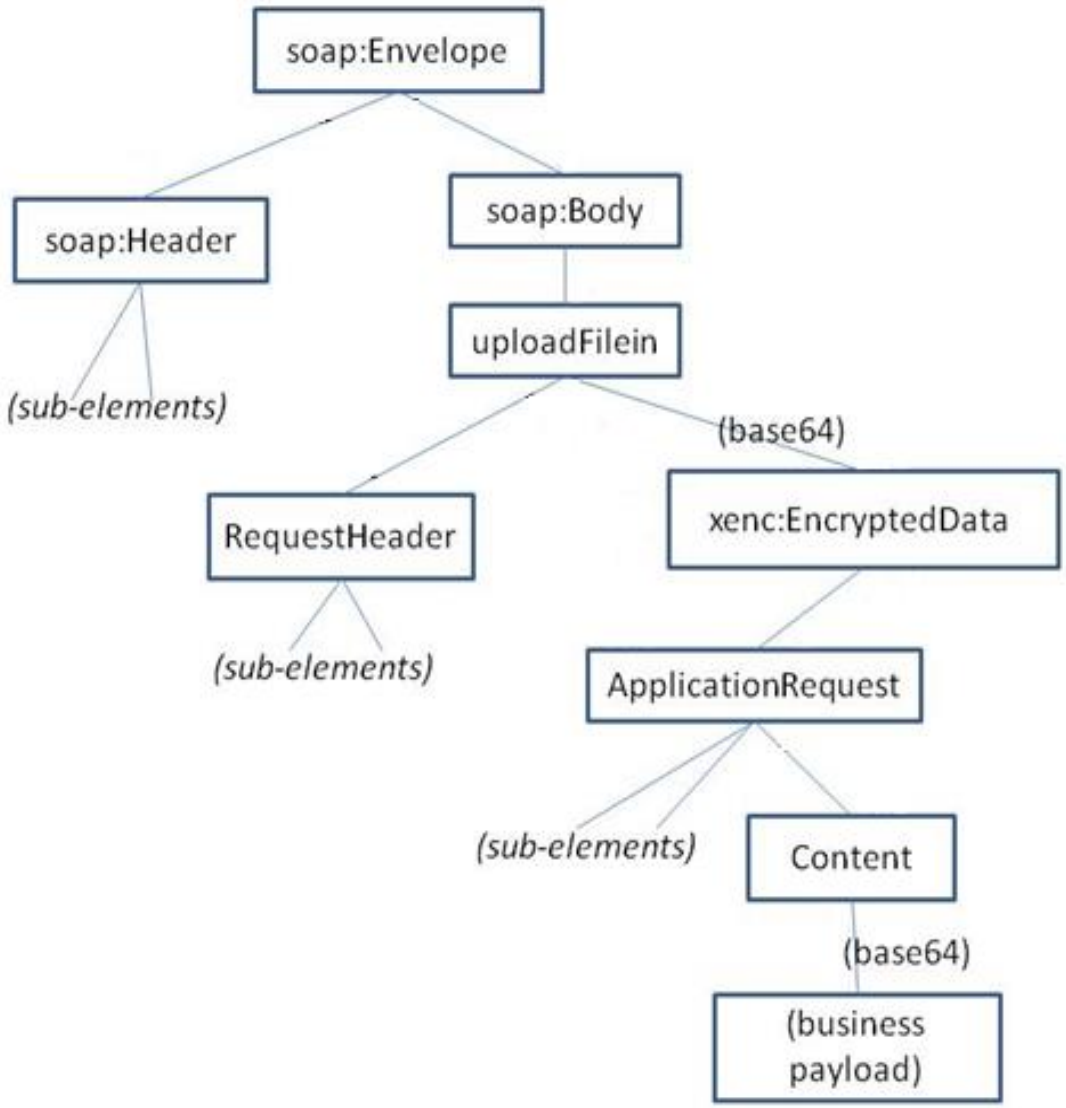
1. The web service call arrives at the server
2. As part of the web service call, the incoming message is checked against the WSDL. This does not include schema checking the ApplicationRequest, due to the base64 encoding.
3. The SOAP header and security information in it is processed, including the transport signature. If everything is not correct, an error message is returned and processing stops.
4. Processing based on RequestHeader can be done.
5. XML is constructed by base64 decoding the content of the ApplicationRequest in the incoming message (in the WSDL, it is defined to be of type xsd:base64Binary). The base64 decoding results in an EncryptedData element.
6. The EncryptedData element is decrypted using XML encryption functionality. This creates an ApplicationRequest element.
7. The ApplicationRequest element can be schema checked now.
8. Further processing of the ApplicationRequest.

The processing of a received ApplicationResponse on the client follows the same pattern. In the same way, the order of processing in the construction of XML prior to a web service call should be as follows:

1. The document for sending is compressed using Gzip
2. The Gzipped file is encoded in base64 so it becomes printable
3. The result is enclosed in a <Content> tag.
4. The so made <Content> tag is placed inside an <ApplicationRequest> tag along with <CustomerId> and other information
5. The Compression is set to true
6. The ApplicationRequest is Digitally signed (business signatures).
7. The ApplicationRequest element is encrypted, producing an EncryptedData element.
8. The EncryptedData element is base64 encoded.
9. A RequestHeader and a soap Body are produced.
10. The base64 data produced in step 8 is inserted in the soap Body in the ApplicationRequest element.
11. The rest of the soap message is constructed (soap Header and soap Envelope)
12. The soap message is digitally signed (transport signature).
13. The Web Service is called.

The construction of an ApplicationResponse on the server follows the same pattern.

The layering of information in the Web Service call is visualized in the following figure:



The figure shows the soap envelope containing the soap Header and soap Body. The soap Body contains an uploadFilein element (the name of this element depends on the operation called, it could also be getUserInfoin, downloadFileListin, downloadFilein, or deleteFilein). This element contains a RequestHeader, and an element containing base64Binary coded data. The base64Binary data can be decoded an EncryptedData element. The EncryptedData element will decrypt into an ApplicationRequest element.

Regarding the Encryption and EncryptionMethod elements

The ApplicationRequest and ApplicationResponse elements contain subelements called *Encryption* (*Encrypted* in ApplicationResponse) and *EncryptionMethod*. These elements can give rise to confusion, since for instance the description of the Encryption elements states:

'If this element is present and the content is the string "true" (case-sensitive) it means that the Content is encrypted or the requested data should be encrypted by the bank'

The first part of the sentence is misleading, since if the Encryption element is visible (i.e. not itself in an encrypted form) then the ApplicationRequest and the Content are no longer encrypted (they may have been in a previous processing step). The only true use of the element is stated in the second part of the sentence: If the element contains "true", the response should be encrypted by the bank. This interpretation also implies that the Encrypted element in ApplicationResponse serves no purpose, since no reply is needed in response to the response.

Similarly, the EncryptionMethod element is only used to communicate the cipher to be used in the response. For this, one EncryptionMethod element is inadequate, since two encryption ciphers are needed: One for symmetric encryption to encrypt the actual data, and one for asymmetric encryption to encrypt the symmetric key. This problem should be solved in a future version of the specification (see the following section). At present it should be up to the individual banks to decide how to handle this. Maybe the same encryption ciphers used in the request should be used in the response, or maybe the same ciphers should always be used.

The specification does not state a default behavior if the Encryption and EncryptionMethod elements are not present in ApplicationRequest.

Regarding the XML Encryption specification

An introduction to the XML Encryption specification is outside the scope of this text. It is recommended to read the specification, which can be found at http://www.w3.org/TR/xmlenc-core/. However, since the specification contains some degrees of freedom as to the XML elements and their content, a brief description of the expected content of an EncryptedData element (which is the root element of encrypted data) will be given here. The encryption used is in two layers:

1. The actual business data is encrypted using symmetric cryptography. The key used to perform this encryption is called the *ephemeral key.* A new ephemeral key is generated every time a message is encrypted. In the example below, the encrypted business data is contained in the EncryptedData/CipherData element. The symmetric cipher is declared in the EncryptedData/EncryptionMethod/@Algorithm attribute.
2. The ephemeral key is encrypted using asymmetric cryptography. The encrypted ephemeral key is contained in the EncryptedData/KeyInfo/EncryptedKey element. The asymmetric cipher is declared in the EncryptedData/KeyInfo/EncryptedKey/EncryptionMethod/@Algorithm attribute. The actual encrypted ephemeral key is contained in the EncryptedData/KeyInfo/EncryptedKey/CipherData element. The certificate used to perform the asymmetric encryption is contained in the EncryptedData/KeyInfo/EncryptedKey/KeyInfo/X509Data/X509Certificate element.

The expected structure of the EncryptedData element can be found in the abbreviated example below.

```
<xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
Type="http://www.w3.org/2001/04/xmlenc#Element">
    <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
    <dsig:KeyInfo xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
        <xenc:EncryptedKey>
            <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
            <dsig:KeyInfo>
```

```
                    <dsig:X509Data>
                        <dsig:X509Certificate>MIIDyT...UijzrQQ==</dsig:X509Certificate>
                    </dsig:X509Data>
                </dsig:KeyInfo>
                <xenc:CipherData>
                    <xenc:CipherValue>NoO3acd...ggGQWA=</xenc:CipherValue>
                </xenc:CipherData>
            </xenc:EncryptedKey>
        </dsig:KeyInfo>
        <xenc:CipherData>
            <xenc:CipherValue>279oO2AZ6...okVnVDvI+KfG</xenc:CipherValue>
        </xenc:CipherData>
</xenc:EncryptedData>
```

Messages sent to the bank must obey the following rules:

- The EncryptedData/KeyInfo/EncryptedKey/KeyInfo/X509Data/X509Certificate element must be present, and it must contain the base64 encoded certificate used to perform the encryption. This is necessary since at one time the bank may have several valid encryption certificates.
- The actual business data must be encrypted using a symmetric cipher. It is not allowed to use asymmetric encryption to encrypt the business data directly. This is because encryption and decryption using asymmetric ciphers is much more CPU intensive than symmetric ciphers.

Clients should use standard software solutions (libraries) supporting the XML encryption specification rather than implementing it on their own.

Encryption Algorithms supported

For incoming requests, the bank supports the following encryption algorithms:

- Symmetric encryption:
  - 3DES. http://www.w3.org/2001/04/xmlenc#tripledes-cbc
  - AES128. http://www.w3.org/2001/04/xmlenc#aes128-cbc
  - AES192. http://www.w3.org/2001/04/xmlenc#aes192-cbc
  - AES256. http://www.w3.org/2001/04/xmlenc#aes256-cbc
- Asymmetric encryption: RSA-v1.5. http://www.w3.org/2001/04/xmlenc#rsa-1_5

Responses from the bank are encrypted with 3DES and RSA-v1.5.

Our implementation of compression

Regarding compression, it should be applied at the Content element level. The Content element contains base64Binary coded data. Depending on the content of the Compression element, the base64Binary data can be in different formats:

1. If the Compression element contains 'true', the data is compressed. For decompression and decoding, the base64Binary data should be converted into binary data, which should be input to the decompression algorithm. The binary output of the decompression algorithm will be the legacy data or SEPA format XML.
2. If the Compression element contains 'false', the data is not compressed. For decoding, the base64Binary data should be base64 decoded. The binary output of this will be the legacy data or SEPA format XML.

When generating compressed data inside the Content element, the following procedure should be used:

The binary legacy data or the SEPA XML is used as input to the compression algorithm. The resulting binary data is base64 encoded and placed inside the Content element. Notice, that the input to the compression algorithm is the **binary** legacy or SEPA XML data, not a base64 encoded version of the data.

Regarding the compression algorithm, gzip (RFC1952) must be used [5]. The CompressionMethod element should contain the string "gzip".

Regarding signatures

Multiple business signatures

The Financial Messages specification [1] allows for up to three enveloped business signatures in the ApplicationRequest element. However, if more than one business signature is to be used, care has to be taken.

If enveloped signatures based on the standard enveloped signature transform (http://www.w3.org/2000/09/xmldsig#enveloped-signature) are used, only the last signature will be verifiable. This is because the standard enveloped signature transform removes only the signature being verified, and leaves other signatures in the element still in place when the message digest is calculated. This effectively means that the message digest value of the signed element is changed every time a signature is added. Thus, the first signature will not be verifiable after the second signature has been added. If a third signature is added, the second signature will not be verifiable either.

The best solution to this problem probably is to not support multiple business signatures until a new version of the specification [1] and the corresponding schemas [4] has been made.

Supported algorithms

For incoming requests, the bank supports digital signatures based on the following algorithms:

- For message digesting,
  - SHA1 (http://www.w3.org/2000/09/xmldsig#sha1), or
  - SHA256 (http://www.w3.org/2001/04/xmlenc#sha256), or
  - SHA512 (http://www.w3.org/2001/04/xmlenc#sha512) should be used.
- For signing,
  - RSA based on SHA1 (http://www.w3.org/2000/09/xmldsig#rsa-sha1), or
  - RSA based on SHA256 (http://www.w3.org/2001/04/xmldsig-more#rsa-sha256), or

    - RSA based on SHA512 (http://www.w3.org/2001/04/xmldsig-more#rsa-sha512) should be used.
- For canonicalization, exclusive canonicalization without comments (http://www.w3.org/2001/10/xml-exc-c14n#) should be used.

Responses from the bank uses the following algorithms:

- For message digesting, SHA1 (http://www.w3.org/2000/09/xmldsig#sha1),
- For signing, RSA based on SHA1 (http://www.w3.org/2000/09/xmldsig#rsa-sha1),
- For canonicalization, exclusive canonicalization without comments (http://www.w3.org/2001/10/xml-exc-c14n#).

Requirements on the signatures

When reading the XMLDSIG standard, there is a lot of liberty in the way a signature can look. In order to be accepted by Danske Bank, every signature must contain the certificate used to create the signature.

For enveloped style signatures (business signatures), this means that a Signature/KeyInfo/X509Data/X509Certificate element must be found. The element must contain the signing certificate in base64 form.

For the WSSEC signatures (transport signatures), it means that a Security/BinarySecurityToken element must be found. The BinarySecurityToken must contain the signing certificate in base64 form, and it must be referenced from the Signature/KeyInfo/SecurityTokenReference element.

See the Appendix for examples of signatures satisfying these requirements.

Additionally, there is a time requirement on the transport signature. The transport signature contains a timestamp indicating at what time the signature was made. The signature will only be accepted if this timestamp is less than 60 minutes old. The interval in which the signature will be accepted is subject to change.

Future changes to the specification

Work has started to revise the financial web-service specification [1] and the corresponding schemas [4] to solve the problems described in this document. Care will be taken to make the changes backwards compatible whenever possible.

References

1. Security and Message Specification For Financial Messages Using Web Services. Nordea, OP-Pohjola Group, Sampo Bank. Version 1.05.
2. EDI Web Services, Danske Bank, [link]
3. PKI Service Description, Danske Bank, [link]
4. The WSDL and schema files for the financial services: BankCorporateFileService_20080616.wsdl, ApplicationRequest_20080918.xsd, ApplicationResponse_20080918.xsd.
5. RFC1952 - GZIP file format specification version 4.3. https://www.ietf.org/rfc/rfc1952.txt

Appendix A: Example XML and SOAP

This appendix will go through the generation of an example signed and encrypted SOAP message. The XML shown here is delivered in a zip-file along with this document.

Generation of the ApplicationRequest

First a business content and ApplicationRequest element must be generated. This corresponds to step 5 on the list on page 5. An example ApplicationRequest is found below:

```
<bxd:ApplicationRequest xmlns:bxd="http://bxd.fi/xmldata/">
      <bxd:CustomerId>ABC123</bxd:CustomerId>
      <bxd:Command>UploadFile</bxd:Command>
      <bxd:Timestamp>2021-12-17T09:30:47Z</bxd:Timestamp>
      <bxd:Environment>TEST</bxd:Environment>
      <bxd:Encryption>true</bxd:Encryption>
      <bxd:Compression>true</bxd:Compression>
      <bxd:CompressionMethod>gzip</bxd:CompressionMethod>
      <bxd:SoftwareId>CustomerSoftwareId</bxd:SoftwareId>
      <bxd:FileType>pain.001.001.02</bxd:FileType>
      <bxd:Content>UjBsR09EbGhjZ0dTQUxNQUFBUUNBRU1tQ1p0dU1GUXhEUzhi</bxd:Content>
</bxd:ApplicationRequest>
```

Note that the business content and the attributes may not make sense in this example.

Signing the ApplicationRequest (business signature)

A business signature is added to the ApplicationRequest. The signature is of the enveloped type. For this the signing certificate of the customer is used. This corresponds to step 6 on the list on page 5:

```
<bxd:ApplicationRequest xmlns:dpstate="http://danskebank.dk/AGENA/SEPA/dpstate"
xmlns:sig="http://danskebank.dk/AGENA/SEPA/SigningService" xmlns:bxd="http://bxd.fi/xmldata/">
      <bxd:CustomerId>ABC123</bxd:CustomerId>
      <bxd:Command>UploadFile</bxd:Command>
      <bxd:Timestamp>2021-12-17T09:30:47Z</bxd:Timestamp>
      <bxd:Environment>TEST</bxd:Environment>
      <bxd:Encryption>true</bxd:Encryption>
      <bxd:Compression>true</bxd:Compression>
      <bxd:CompressionMethod>gzip</bxd:CompressionMethod>
      <bxd:SoftwareId>CustomerSoftwareId</bxd:SoftwareId>
      <bxd:FileType>pain.001.001.02</bxd:FileType>
      <bxd:Content>UjBsR09EbGhjZ0dTQUxNQUFBUUNBRU1tQ1p0dU1GUXhEUzhi</bxd:Content>
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
<SignedInfo>
  <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
  <SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/>
  <Reference URI="">
    <Transforms>
      <Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
      <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
    </Transforms>
    <DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
    <DigestValue>Kg6GoK0HGEXcOiziLjFbg3mw8D6esmwNWbU9qTFRTkM=</DigestValue>
  </Reference>
</SignedInfo>
```

<SignatureValue>HzteEuDTV/kFHMfa3IACietPwxZexBaFtykoav34/+JeyqjshRK5nDPMguBfgDUy+K41dbUiI
lGrB8WKoV1etiQga2JT3SsDj2Os+h7Ov/87SNSQHb8I35BTfiX3ykKlNjIEuNjuc1fwRlWIUszctpVtuOMtrpi+1d
+0ujo9kKHhen+NKv19PR/+DUWnqwDBEffHV7msAkFDSHobmwmt3zj96XBpPahMlg/FtyzLurphxA8ddUSSk
CbEkVhGrHAj30cA+3dC5JXvlwy40sNv4tHUgg1cAaRBwGjHwsKKI/k9D2EfCqthq2juXXk7iX0GXf7GEqfslDF
LBLNvJTEEqw==</SignatureValue><KeyInfo><X509Data><X509Certificate>MIIDIjCCAougAwIBAgIEL4j5xz
ANBgkqhkiG9w0BAQUFADBpMQswCQYDVQQGEwJESzEQMA4GA1UEBxMHRGVubWFyazEaMBgGA1U
EChMRRGFuc2tlIEJhbmsgR3JvdXAxEDAOBgNVBAsTB0FwcGxTSW4xGjAYBgNVBAMUEUZpbm5pc2hfU
0VQQV90ZXN0MB4XDTA5MTAyNjEyMDMzMVoXDTEyMDIwMzEyMDMzMVowaTELMAkGA1UEBhMCRE
sxEDAOBgNVBAcTB0Rlbm1hcmsxGjAYBgNVBAoTEURhbnNrZSBCYW5rIEdyb3VwMRAwDgYDVQQLEwd
BcHBsU0luMRowGAYDVQQDFBFGaW5uaXNoX1NFUEFfdGVzdDCBnzANBgkqhkiG9w0BAQEFAAOBjQA
wgYkCgYEAyKLcdWd6TvZ+AF3zAabivvf+s+QoTnxQ4+DOl8mDBMpo7Ka41i59u79qyHa243HuQgW4eCMi
st+CUfY/Ynu6DgsP/NFduuV2VZNMYdQIKwJMaytQ1zhYufMeiw8YdgGk8qVjZJOmwbCjnDI1+da0CvEeEs1
ZUZtVK9nRDDsnG9cCAwEAAaOB1jCB0zAMBgNVHRMEBTADAQH/MB0GA1UdDgQWBBRu/JyaNnOeCQ
C6jNdNZl/qfLNoYTCBlgYDVR0jBIGOMIGLgBRu/JyaNnOeCQC6jNdNZl/qfLNoYaFtpGswaTELMAkGA1UEB
hMCREsxEDAOBgNVBAcTB0Rlbm1hcmsxGjAYBgNVBAoTEURhbnNrZSBCYW5rIEdyb3VwMRAwDgYDV
QQLEwdBcHBsU0luMRowGAYDVQQDFBFGaW5uaXNoX1NFUEFfdGVzdIIEL4j5xzALBgNVHQ8EBAMCAr
wwDQYJKoZIhvcNAQEFBQADgYEAq6rHlg41XYLMjOLZ33toWKLos3uWJucHK0F6TKAXSNp7Qo34hx04P
0MidOXuj18mYZkU/usVHn/L6rkWSfJretpO0xWTzw50XSm83JUQIYeQyMiU/w7ByJSNlTa8G1i4nKWjZ/kcl2
Ur0oDUTTf/X8mlC24EkDCl9RQjGEYdUvE=</X509Certificate><X509IssuerSerial><X509IssuerName>CN=
Finnish SEPA test, OU=ApplSIn, O=Danske Bank Group, L=Denmark,
C=DK</X509IssuerName><X509SerialNumber>797505991</X509SerialNumber></X509IssuerSerial></X50
9Data></KeyInfo></Signature></bxd:ApplicationRequest>

Encrypting the ApplicationRequest

The signed ApplicationRequest element is encrypted using XML encryption. For this, the encryption certificate of the bank is used. This corresponds to step 7 on the list on page 5:

<xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
Type="http://www.w3.org/2001/04/xmlenc#Element">
<xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
<dsig:KeyInfo xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
<xenc:EncryptedKey>
<xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
<dsig:KeyInfo>
<dsig:X509Data>

<dsig:X509Certificate>MIIDyTCCArGgAwIBAgIKNDQ0NDQzMDAwMjANBgkqhkiG9w0BAQsFADCBxD
EQMA4GA1UEAxMHREJHUk9PVDELMAkGA1UEBhMCREsxEzARBgNVBAcTCkNvcGVuaGFnZW4xEDA
OBgNVBAgTB0Rlbm1hcmsxGjAYBgNVBAoTEURhbnNrZSBCYW5rIEdyb3VwMRowGAYDVQQLExFEYW5
za2UgQmFuayBHcm91cDEYMBYGA1UEBRMPNjExMjYyMjgxMTMwMDAxMQkwBwYDVQQEEwAxCTAHB
gNVBCoTADEJMAcGA1UEDBMAMQkwBwYDVQQREwAwHhcNMDkxMDMwMTIwMDAwWhcNMTExMDM
wMTIwMDAwWjCBxTERMA8GA1UEAxMIREJHQ1JZUFQxCzAJBgNVBAYTAkRLMRMwEQYDVQQHEwpD
b3BlbmhhZ2VuMRAwDgYDVQQIEwdEZW5tYXJrMRowGAYDVQQKExFEYW5za2UgQmFuayBHcm91cDE
aMBgGA1UECxMRRGFuc2tlIEJhbmsgR3JvdXAxGDAWBgNVBAUTDzYxMTI2MjI4NDQzMDAwMjEJMAcG
A1UEBBMAMQkwBwYDVQQqEwAxCTAHBgNVBAwTADEJMAcGA1UEERMAMIGdMA0GCSqGSIb3DQEB
AQUAA4GLADCBhwKBgQCpae2tNbCmPRni74t4pXpzUwsUTRBeazPpchZHjn+HBpfQAH66kCzOcxLHOrL
71CFYIj6B4WrqXZx4SUldQpKu67vSh5ZsLyyXPZniCo+DtX2zOYGEigzq6hIylR5XKnXkZfJSxo/TgGX3rkLxF
6cbaP6mxN7LcCv9dUj+eRnngQIBA6NAMD4wEQYDVR0OBAoECERCR0NSWVBUMBkGA1UdIwQSMBC
ADsLGxMP29sL29/X39cXEMA4GA1UdDwEB/wQEAwIEMDANBgkqhkiG9w0BAQsFAAOCAQEAD8+BpnM/
K10m5iKO4BTcvGi77oSp0sul5utP6ZFEveuqjlMx4xxh+1SVRSMXQttd49/08BIUOaYrljvP+dJT5J6D+UeFRSj
E6mevjRE1qqIpCsHoCWVE9WkiuF3QLaouD9PH5FPsk53QVdd4Yp+vIKDVYQbpc12A8wAEu8hkngqxN2/E
pd8u/3LQgsTmY6mn5ek2EFNI6o2dQa503DpzjoXGURIFOoAaWB7ZUJhe7nxG1Xjn1OnttreoP6twCTrG8IoL
fGKDkLVQ3G+7vAqXgB+JwFLUoULRaw0eISPgE7nMkz0la0lu+aJlaHRJZq/F9eMkDXFqRSMGHHNUijzrQ
Q==</dsig:X509Certificate>

```
                    </dsig:X509Data>
                </dsig:KeyInfo>
                <xenc:CipherData>

    <xenc:CipherValue>NoO3acdWBuH0nduRMdLZtOhN645erE7JjABu4waNg7GYbEvxrj6m1mdhpj32BZ4
el+cp5r5UlhEYxt4sHlj8tMxJqcogxrJEiD6Ig/zzc1mDV61fQidJWgclkWOUA2deb1Wz4OeS1oMe1F6M1wmvS
PWNpSZbEmEG/4zv9ggGQWA=</xenc:CipherValue>
                </xenc:CipherData>
            </xenc:EncryptedKey>
        </dsig:KeyInfo>
        <xenc:CipherData>

    <xenc:CipherValue>279oO2AZ63oVYSFjMT+mW/YSUBbWJmNiGeg1OlendoFEhhpHYcpTbGXG7kO/
SXtewRr1aZp19L4/1qqNJW0BQeXluwVd7Qz9odekyzIGN4qBPuvp3ZH46yPT7kPfc3umwUIVZ5R3GPhEYG
m4AUhK8Ygju/FZPZy+iU84bUG5iSV8ORYtEck+UQpGbbfCyYPX5U96zkV0hj6HL+Cwelg2Q9Gp6chu0IpBL
/WRF0n1Kn1J/Z7LkVbKB9cROm3B2v77vJkjrhq4FwwT0wlkZZbq4yX5w7qeMlgvAUZ2OdPDwhEGSj+TM6ai
dGYddNDhQ0SDfbLKETWg8Lh6vuRhFk925uVUoiLJgw1MBf22t3H1Tz5z30leOJHdsEJt6XfHP0P2WUpSw1
EAAqFq4RBl4hOgrzoRqVfZM9u1MPML8bYhLDl4O8cKCDvo4iwd1Mv0niu0veD7Asyo43nA4hVh/BsUskEZe
/SaStOo+WXzeL7cdJmeW9AkTkpsiidF8iEERY2iANdJpDNGS7B7jQ3uDZiVOVYBoCG2YrecMSrlcpfxzNap
DJKUshyxRigfPSGtN9YkMZsgc/whhklVN1VnV2A541butR9SG3m8lTbm/fgcvt36DPXcmlV4cyHWTiX2lv9LF
+lAmG+WVBnu6d4LMCXWMa4gzTnrgoXvXpAKNl51UmH6jiYOn1Or8Ls5YOzDeZRcVdgRFT1K2sHrZ4W
WYeSUQfsxfZo9qzngTQ4BvAomPrL6Xh4lYAPMQxJZio1yo8/C7lyZiwv/XAeNuZ4/tG7tF3vKi/B6mjHm9iqN6
OtAuQ8iHCZt8ayBMhbgHb6vORxv77A1+OKxCkZQfcxyHesCnjj0ONEvnoJi3Z/2TjupjAMYNgJ+O20Wt4Q8J
hWGfuCOl13H4oy2Z/Wy+lsKqRyJ4ihlXrz0T5LsFwncdE7ho3MhyT2C3NAkrosgpVRlyB99DLJpWhWR4s3tx
pfc47DRihny3gq7iepcl60ANTL1tBePk0PRD/y/tf59aZjhWSMBWQpvwJaVAHJgpBGtCUYkGXLAkUICZ+cY4
osHeTylivNshX4F/O8AvDjXvlQUKYss+Bz7i+raQMFVvQNp0Lr3BLc3FPAQCgZDFB9xuaVOu245hxtWakaq5
ZC/jg09KkA+Vwj13nkK0dY0YE3IDIxTIEVvuN61ulRJEBwh0JWU96fPycVfmhdvhlQF6lcWzvbsCKhAMZu83v
0Wb941LsssZzlKHa7Qj/2U4maTHgl03iZhdETQtVyvrd1FS+PK86V62jDhy53VSF1bqP9eH/16bqLXSSyvyCE
lLEO4LhZCbS33fUpKM0sej5gp7AQhFTA4kVrv0K6D9/7R0gR72GZF9/+HXwKq4VHg+s8380xen3F5O9ZWe
2nX9y3yIeH2Us/PoKjyYjKwV3DWYUi0Jtvw93uobzxAMLFhqFPX8+bHFbnyN1yqEjJFB5BZkX8Xa1G8Sp6ig
kGtXZltX0laxBzrWvbpyuZW2LU/1igkQkA20cuLFPrEax1gZzVZyBP9zEKdqCGRUGc+HFuirgtG0ncR+4KKbf
VtN7r3xkVNfaihxXVeF45T1/TBE+mdjRd5kMICLEY29uT3IFZbZsPUoSSOEa6wNLYtQTSt7DSJiX4abhxUDx
QeHgF0iLbKJw99h/OjopfOCPyTIlxglayLIT6TrDsiwHnxuwbRkeRmh7gXS79qBZlyBpi1g6ZMTB4FDfP0+1QI
FzT5wmoOlzEZ1SgOT8JgGd5c/xeLZ3NqLvlO4ZxefO6KV+l8IadOipV5utV3mOhgAl9xDzOnqYb7UZy9OcZR
p9hWX6KxzokyomQGwUqjAio/k3wqpwGBuCNgLd+piWUBkot2RRlQeVFeYjaa0hTsNmAmpbvgoj2HhERFfg
RQk6/IU4XbBSHmSX3Va+NK56uDEBqPhEg57G6Ok1ozqXQPRJayob87GZaEzEgUDW12ieXYIjXXZIz92Ie
w04/2bhXc8dtQEyxOcNkFugfWEd7EeNyFULSAljdsQfXzaouOKsWL3XzQ71+sIIJkFU0KfKitxlFaukmwUfLx
W/ZwGgIQcOchs7E5Tu5ZGil+ih5wOm1py+M+cgBk3wyFGlcjD9IJZ5oM9IQAjJn78NwTQ8LG8VrFzAe0ObM
7/8sWZbr+c1lMmDvAI3qAARray6rNTMamqgu0cc1hcdyEyn3CmZjy1x9VWbbNXdqk/lXu6c6Kgda8rMrVyV9
B9KyWutw9ht9u8bK9ny31zDnIlzEm3sVaY4XzYbXqvX7ZpVvcC607KiKE7m9HWcuPXsqA7eFPbPrZl6Jue9
F8OkZIIH4vsOY+FVo0ju9JufJSoAA+eHctUZ4c4biYPJlxa82THQT9XkE47X8eWF2ucY6upGJ8La8H27wavM
cuKJNYxMnJKKpdpMlmBMgTZuyvyAzfoyO/CiaSggm3wanYl586FNe9zXnDqtsR+4RTNrBNd28fqwyneLoeEi
3Om/dOB3WLGQHkCyburz3HIn8ZPOr2dDMuRQzs8WBMr2KUohdQ6jXonYEEhKFeJUx2HFamPgjKl+Mcp
+6a3as3L8I10JihBdgrM6vKe7fHc6dPhzNVz3ep8AYll/6sr01v5uAK42Oy9Q9TNxZsL47zecMd4CwRJdHQMS
4DKfiHZBOLE3Cep7Ca5bKhYlkTK+nkCw5twkMSJdqzEqaVu0WPFwLQv4ysibrOPtN5E+alzWYQPnGB2kcq
3vH0mfqnxcP1aLxJBtR0u5MPWehfCem9i/lO/few4XnbplMMcf8YjzH1brpp+P667c9O9qVDAsj9KFVlVZlpv2Z
0DU3pS49oZkU+vVQWOahFqXYYNCbW5kJhXwAjHFjHTjQnRQPcGqkjJ4fEiLvYOQ5yE1hW/LwNdua8NcC
nzuDg6P7SpOmIZ8p3GENITvPRUxZvTcDK5efdsoVb8FXVuyftVxwUXmnZ9FCmdjwr57tt3iWb8S064LHJYO
6X4HvcfR4VtPBVjVO5qCE57hdy+XYOvylT9bhCWLvdlI354ni6L6lgxx0Y/fiBNZlmT7FeVl91PIw6lW0qvzHR2
Pkz+abzvqU5pNlTyJRt20+5hezvwnL//VTLyZmBENtiG6FL4yi1NL+PWcAvVw86kQBlkTvlSuMvaZwQ1kRIrw
1gu/uEGcU5COacYwvKTuL/GMh10g5e0GYcYEdHCtjTE5CCWCOYt/tTLgNkeV2Q7QdY5Xk9Zsc3zS1Krwr4
Dbs9Hk4WCWHhoM4ZUn35F08bs2p2N0bdeag1BQ1Wn/tHdJ3l2ZbpAqJGkZYj4BcEW+3oI9VmaQOI+eIGN
Odv4lydlOtoz8bL50LqxaCMGIAu5BCQUHRZKOm95SvcLZocG+UCPKnrfVXkCPz20nhwMXh6ZPK/vBCoE
Yf2ICSMJfpvEQRFKeM/j4SYSjm/C0C3QneeCtOjxmCrwqinKqrzkmMpNTYXZjUua3WtGjq4H4nvwVHKGSL/
ZSs6OifBO3YBFEYcHqC58ZtYGBafFH+aaTdNQqMpd8X0U37rgtqfeg06NSt+QT9Ezj35SL7yjs+4+rvX2Rpyk
FD3s4S2l+3aVBxqY5vTTlKP7iDZH1ADKjn7rSxeiuJYhxqJnNzQyMMsgGmYbW3rnjlsTqKU/XD6twOcEb17K
SII9+2Jp3MGjcLPdYdWKsiobdamdU9Cj6mIhG5d1DThNajwdTXjVzP7aPsmjJtNaMPFRr2l62wSiUFtLSlLJm
pF9W98umQyWpfWXJ9yOx1zCR674X4JEiT6waP1cGV7ZGklmYbr/Q3XSfjJAMKZvqOpnVLpzZwOGVq4oQ
```

W9DRFEL6wy3Ax2yHywykdqUeA2TFY8eo47X3WAO9lyX9ZPAS2vaJLB/XHlavOHc3p3OWT3IGpq6ngsO2
1FokVnVDvI+KfG</xenc:CipherValue>
</xenc:CipherData>
</xenc:EncryptedData>

This step should be performed by a piece of standard software.

Generation of the SOAP message

The next step is to generate a SOAP message (including the RequestHeader), base64 encode the XML from the previous step and insert it in a the ApplicationRequest element of the SOAP. This corresponds to steps 8 to 11 on the list on page 5:

```
<soapenv:Envelope xmlns:sign="http://danskebank.dk/AGENA/SEPA/SigningService"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:cor="http://bxd.fi/CorporateFileService"
xmlns:mod="http://model.bxd.fi" xmlns:dpstate="http://danskebank.dk/AGENA/SEPA/dpstate">
    <soapenv:Header/>
    <soapenv:Body>
        <cor:deleteFilein>
      <mod:RequestHeader>
        <mod:SenderId>ABC123</mod:SenderId>
        <mod:RequestId>33</mod:RequestId>
        <mod:Timestamp>2021-12-17T09:30:47Z</mod:Timestamp>
        <mod:Language>EN</mod:Language>
        <mod:UserAgent>CustomerSoftwareId</mod:UserAgent>
        <mod:ReceiverId>DABAFIHH</mod:ReceiverId>
      </mod:RequestHeader>
          <mod:ApplicationRequest>
PHhlbmM6RW5jcnlwdGVkRGF0YSB4bWxuczp4ZW5jPSJodHRwOi8vd3d3LnczLm9yZy8yMDAxLzA0L3htb
GVuYyMiIFR5cGU9Imh0dHA6Ly93d3cudzMub3JnLzIwMDEvMDQveG1sZW5jI0VsZW1lbnQiPjx4ZW5jOkVu
Y3J5cHRpb25NZXRob2QgQWxnb3JpdGhtPSJodHRwOi8vd3d3LnczLm9yZy8yMDAxLzA0L3htbGVuYyNhZ
XMyNTYtY2Jji8+PGRzaWc6S2V5SW5mbyB4bWxuczpkc2lnPSJodHRwOi8vd3d3LnczLm9yZy8yMDAwLzA
5L3htbGRzaWcjIj48eGVuYzpFbmNyeXB0ZWRLZXk+PHhlbmM6RW5jcnlwdGlvbk1ldGhvZCBBbGdvcml0a
G09Imh0dHA6Ly93d3cudzMub3JnLzIwMDEvMDQveG1sZW5jI3JzYS0xXzUiLz48ZHNpZzpKZXlJbmZvPjxk
2lnOlg1MDlEYXRhPjxkc2lnOlg1MDlDZXJ0aWZpY2F0ZT5NSUlEeVRDQ0FyR2dBd0lCQWdJS05EUTBORF
F6TURBd01qQU5CZ2txaGtpRzl3MEJBUXNGQURDQnhERVFNQTRHQTFVRUF4TUhSRUpIVWs5UFZER
UxNQWtHQTFVRUJoTUNSRXN4RXpBUkJnTlZCQWNUQ2tOdmNHVnVhR0ZuWlc0eEVEQU9CZ05WQkF
nVEIwUmxibTFoY21zeEdqQVlCZ05WQkFvVEVUmhibk5yWlNCQ1lXNXJJRWR5YjNWd01Sb3dHQVlEVlF
RTEV4RkVZVzV6YTJVZ1FtRnVheUJIY205MWNERVlNQllHQTFVRUJSTVBOakV4TWpZeU1qZ3hNVE13T
URBeE1Ra3dCd1lEVlFRRUV3QXhDVEFIQmdOVkJDb1RBREVKTUFjR0ExVUVEQk1BTVFrd0J3WURWU
VFSRXdBd0hoY05NRGt4TURNd01USXdNREF3V2hjTk1URXhNRE13MTVRJd01EQXdXakNCeFRFUk1BOE
dBMVVFQXhNSVJFSkhRMUpaVUZReEN6QUpCZ05WQkFZVEFrUkxNUk13RVFZRFZRUUhFd3BEYjNCb
GJtaGhaMlZ1TVJBd0RnWURWUVFJRXdkRVpXNXRZWEpyTVJvd0dBWURWUVFLRXhGRVlXNXphMlVn
UW1GdWF5QkhjbTkxY0RFYU1CZ0dBMVVFQ3hNUlJHRnVjMnRsSUVKaGJtc2dSM0p2ZFhBeEdEQVdCZ
05WQkFVVER6WXhNVEkyTWpJNE5EUXpNREF3TWpFSk1BY0dBMVVFQkJNQU1Ra3dCd1lEVlFRcUV3
QXhDVEFIQmdOVkJBd1RBREVKTUFjR0ExVUVFUk1BTUlHZE1BMEdDU3FHU0liM0RRRUJBUVVBQTRH
TEFEQ0Job0tCZ1FDcGFlMnROYkNtUFJuaTc0dDRwWHB6VXdzVVRSQmVheIBwY2haSGpuK0hCcGZRQ
Ug2NmtDek9jeExIT3JMNzFDRllJajZCNFdycVhaeDRTVWxkUXBLdTY3dlNoNVpzTHl5WFBabmlDbytEdFgy
ek9ZR0VpZ3pxNmhJeWxSNVhLblhrWmZKU3hvL1RnR1gzcmtMeEY2Y2JhUDZteE43TGNDdjlkVWorZVJub
mdRSUJBNk5BTUQ0d0VRWURWUjBPQkFvRUNFUkNSME5TV1ZCVU1Ca0dBMVVkSXdRU01CQ0FEc0x
HeE1QMjIzTDI5L1gzOWNYRU1BNEdBMVVkRHdFQi93UUVBd0lFTURBTkJna3Foa2lHOXcwQkFRc0ZBQ
U9DQVFFQUQ4K0Jwbk0vSzEwbTVpS080QlRjdkdpNzdvU3Awc3VsNXV0UDZaRkV2ZXVxamxNeDR4eGgr
MVNWUlNNWFF0dGQ0OS8wOEJJVU9hWXJJanZQK2RKVDVKNkQrVWVGUlNqRTZtZXZqUkUxcXFJcEN
zSG9DV1ZFOVdraXVGM1FMYW91RDlQSDVGUHNrNTNRVmRkNFlwK3ZJS0RWWVFicGMxMkE4d0FFdT
hoa25ncXhOMi9FcGQ4dS8zTFFnc1RtWTZtbjVlazJFRk5JNm8yZFFhNTAzRHB6am9YR1VSSUZPb0FhV0I
3WlVKaGU3bnhHMVhqbjFPbnR0cmVvUDZ0d0NUckc4SW9MZkdLRGtMVlEzRys3dkFxWGdCK0p3RkxVb1
```

VMUmF3MGVJU1BnRTduTWt6MGxhMGx1K2FKbGFlUkpacS9GOWVNa0RYRnFSU01HSEhOVWlqenJRU
T09PC9kc2lnOlg1MDlDZXJ0aWZpY2F0ZT48L2RzaWc6WDUwOURhdGE+PC9kc2lnOktleUluZm8+PHhlbm
M6Q2lwaGVyRGF0YT48eGVuYzpDaXBoZXJWYWx1ZT5Ob08zYWNkV0J1SDBuZHVSTWRMWnRPaE42
NDVlckU3SmpBQnU0d2FOZzdHWWJFdnhyajZtMW1kaHBqMzJCWjRlSStjcDVyNVVJaEVZeHQ0c0hJajh0T
XhKcWNvZ3hySkVpRDZJZy96emMxbURWNjFmUWlkSldnY0lrV09VQTJkZWIxV3o0T2VTMW9NZTFGNk0x
d212U1BXTnBTWmJFbUVHLzR6djlnZ0dRV0E9PC94ZW5jOkNpcGhlclZhbHVlPjwveGVuYzpDaXBoZXJE
XRhPjwveGVuYzpFbmNyeXB0ZWRLZXk+PC9kc2lnOktleUluZm8+PHhlbmM6Q2lwaGVyRGF0YT48eGVuY
zpDaXBoZXJWYWx1ZT4yNzlvTzJBWjYzb1ZZU0ZqTVQrbVcvWVNVQmJXSm1OaUdlZzFPbGVuZG9GRW
hocEhZY3BUYkdYRzdrTy9TWHRld1JyMWFacDE5TDQvMXFxTkpXMEJRZVhsdXdWZDdRejlvZGVreXpJR
040Z0JQdXZwM1plNDZ5UFQ3a1BmYzN1bXdVSVZaNVIzR1BoRVlHbTRBVWhLOFlnanUvRlpQWnkraVU4
NGJVRzVpU1Y4T1JZdEVjaytVUXBHYmJmQ3lZUFg1VTk2emtWMGhqNkhMK0N3ZWxnMlE5R3A2Y2h1M
ElwQkwvV1JGMG4xS24xSi9aN0xrVmJLQjljUk9tM0lydjc3dkpranJocTRGd3dUMHdJa1paYnE0eVg1dzdxZU
1sZ3ZBVVoyT2RQRHdoRUdTaitUTTZhaWRHWWRkTkRoUTBTRGZiTEtFVFdnOExoNnZ1UmhGazkyNXV
WVW9pTEpndzFNQmYyMnQzSDFUejV6MzBsZU9KSGRzRUp0NlhmSFAwUDJXVXBTdzFFQUFxRnE0Uk
JsNGhPZ3J6b1JxVmZaTTl1MU1QTUw4YlloTERJNE84Y0tDRHZvNGl3ZDFNdjBuaXUwdmVEN0FzeW80M
25BNGhWaC9Cc1Vza0VaZS9TYVN0T28rV1h6ZUw3Y2RKbWVXOUFrVGtwc2lpZEY4aUVFUlkyaUFOZEp
wRE5HUzdCN2pRM3VEWmlWT1ZZQm9DRzJZcmVjTVNybGNwZnh6TmFwREpLVXNoeXhSaWdmUFNHd
E45WWtNWnNnYy93aGhrSVZOMVZuVjJBNTQxYnV0UjlTRzNtOElUYm0vZmdjdnQzNkRQWGNtbFY0Y3lI
V1RpWDJsdjlMRitJQW1HK1dWQm51NmQ0TE1DWFdNYTRnelRucmdvWHZYcEFLTkk1MVVtSDZqaVlPbj
FPcjhMczVZT3pEZVpSY1ZkZ1JGVDFLMnNIclo0V1dZZVNVUWZzeGZabzlxem5nVFE0QnZBb21Qckw2W
Gg0SVlBUE1ReEpaaW8xeW84L0M3bHlaaXd2L1hBZU51WjQvdEc3dEYzdktpL0I2bWpIbTlpcU42T3RBdVE
4aUhDWnQ4YXlCTWhiZ0hiNnZPUnh2NzdBMStPS3hDa1pRZmN4eUhlc0NuamowT05Fdm5vSmkzWi8yVG
p1cGpBTVlOZ0orTzlwV3Q0UThKaFdHZnVDT2wxM0g0b3kyWi9XeStsc0txUnlKNGlobFhyejBUNUxzRnduY
2RFN2hvM01oeVQyQzNOQWtyb3NncFZSSXlCOTlETEpwV2hXUjRzM3R4cGZjNDdEUmlobnkzZ3E3aWVw
Y2w2MEFOVEwxdEJlUGswUFJEL3kvdGY1OWFaamhXU01CV1FwdndKYVZBSEpncEJHdENVWWtHWEx
Ba1VJQ1orY1k0b3NIZVR5SWl2TnNoWDRGL084QXZEalh2bFFVS1IzcytCejdpK3JhUU1GVnZRTnAwTHIz
QkxjM0ZQQVFDZ1pERkI5eHVhVk91MjQ1aHh0V2FrYXE1WkMvamcwOUtrQStWd2oxM25rSzBkWTBZRTN
JREl4VElFVnZ1TjYxdUlSSkVCd2gwSldVOTZmUHljVmZtaGR2aGxRRjZsY1d6dmJzQ0toQU1adTgzdjBXYjk
0MUxzc3NaekILSGE3UWovMlU0bWFUSGdJMDNpWmhkRVRRdFZ5dnJkMUZTK1BLODZWNjJqRGh5NT
NWU0YxYnFQOWVlLzE2YnFMWFNTeXZ5Q0VsTEVPNExoWkNiUzMzZlVwS00wc2VqNWdwN0FRaEZUQ
TRrVnJ2MEs2RDkvN1IwZ1I3MkdaRjkvK0hYd0txNFZIZytzODM4MHhlbjNGNU85WldlMm5YOXkzeUllSDJVc
y9Qb0tqeVlqS3dWM0RXWVVpMEp0dnc5M3VvYnp4QU1MRmhxRlBYOCtiSEZibnlOMXlxRWpKRkI1QlprW
DhYYTFHOFNwNmlna0d0WFpsdFgwSWF4QnpyV3ZicHl1WlcyTFUvMWlna1FrQTlwY3VMRlByRWF4MWd
aelZaeUJQOXpFS2RxQ0dSVUdjK0hGdWlyZ3RHMG5jUis0S0tiZlZ0TjdyM3hrVk5mYWloeFhWZUY0NVQxL
1RCRSttZGpSZDVrTUlDTEVZMjI1VDNJRlpiWnNQVW9TU09FYTZ3TkxZdFFUU3Q3RFNKaVg0YWJoeFVE
eFFlSGdGMGlMYktdKdzk5aC9Pam9wZk9DUHlUSUl4Z2xheUxsVDZUckRzaXdlbnh1d2JSa2VSbWg3Z1hTN
zlxQlpJeUJwaTFnNlpNVEl0RkRmUDArMVFJRnpUNXdtb09sekVaMVNnT1Q4SmdHZDVjL3hlTFozTnFMdm
xPNFp4ZWZPNktWK2w4SWFkT2lwVjV1dFYzbU9oZ0FJOXhEek9ucVliN1VaeTlPY1pScDloV1g2S3h6b2t5b
21RR3dVcWpBaW8vazN3cXB3R0J1Q05nTGQrcGlXVUJrb3QyUlJsUWVWRmVZamFhMGhUc05tQW1wYn
Znb2oySGhFUkZmZ1JRazYvSVU0WGJCU0htU1gzVmErTks1NnVERUJxUGhFZzU3RzZPazFvenFYUVBS
SmF5b2I4N0daYUV6RWdVRFcxMmllWFlJalhYWkl6OTJJZXcwNC8yYmhYYzhkdFFFeXhPY05rRnVnZldFZ
DdFZU55RlVMU0FsamRzUWZYemFvdU9Lc1dMM1h6UTcxK3NJSUprRlUwS2ZLaXR4bEZhdWttd1VmTHh
XL1p3R2dsUWNPY2hzN0U1VHU1WkdpSStpaDV3T20xcHkrTStjZ0JrM3d5RkdsY2pEOUlKWjVvTTlJUUFq
Sm43OE53VFE4TEc4VnJGekFlME9iTTcvOHNXWmJyK2MxbE1tRHZBSTNxQUFScmF5NnJOVE1hbXFnd
TBjYzFoY2R5RXluM0NtWmp5MXg5VldiYk5YZHFrL2xYdTZjNktnZGE4ck1yVnlWOUI5S3lXdXR3OWh0OXU
4Yks5bnkzMXpEbklsekVtM3NWYVk0WHpZYlhxdlg3WnBWdmNDNjA3S2lLRTdtOUhXY3VQWHNxQTdIRlBi
UHJabDZKdWU5RjhPa1pJSUg0dnNPWStGVm8wanU5SnVmSlNvQUErZUhjdFVaNGM0YmlZUEpJeGE4Ml
RlUVQ5WGtFNDdYOGVXRjJ1Y1k2dXBHSjhMYThlMjd3YXZNY3VLSk5ZeE1uSktLcGRwTWxtQk1nVFp1eX
Z5QXpmb3lPL0NpYVNnZ20zd2FuWWw1ODZGTmU5elhuRHF0c1IrNFJUTnJCTmQyOGZxd3luZUxvZUVp
M09tL2RPQjNXTEdRSGtDeWJ1cnozSEluOFpQT3lyZERNdVJRenM4V0JNcjJLVW9oZFE2alhvblFRWhLR
mVKVXgySEZhbVBnaktsK01jcCs2YTNhczNMOEkxMEppaEJkZ3JNNnZLZTdmSGM2ZFBoek5WejNlcDhB
WWxsLzZzcjAxdjV1QUs0Mk95OVE5VE54WnNMNDd6ZWNNZDRDd1JKZEhRTVM0REtmaUhaQk9MRTN
DZXA3Q2E1YktoWUlrVEsrbmtDdzV0d2tNU0pkcXpFcWFWdTBXUEZ3TFF2NHlzaWJyT1B0TjVFK2FseldZ
UVBuR0lya2NxM3ZIMG1mcW54Y1AxYUx4SkJ0UjB1NU1QV2VoZkNlbTlpL2xPL2ZldzRYbmJwbE1NY2Y4
WWp6SDFicnBwK1A2NjdjOU85cVZEQXNqOUtGVklWWmxwdjJaMERVM3BTNDlvWmtVK3ZWUVdPYWhG
cVhZWU5DYlc1a0poWHdBakhGakhUalFuUlFQY0dxa2pKNGZFaUx2WU9RNXlFMWhXL0x3TmR1YThOY0
NuenVEZzZQN1NwT21JWjhwM0dFTklUdlBSVXhadlRjREs1ZWZkc29WYjhGWFZ1eWZ0Vnh3VVhtblo5RkN
tZGp3cjU3dHQzaVdiOFMwNjRMSEpZTzZYNEh2Y2ZSNFZ0UEJWalZPNXFDRTU3aGR5K1hZT3Z5SVQ5Y

```
mhDV0x2ZGxJMzU0bmk2TDZsZ3h4MFkvZmlCTlpsbVQ3RmVWbDkxUEl3NklXMHF2ekhSMlBreithYnp2cV
U1cE5sVHlKUnQyMCs1aGV6dnduTC8vVlRMeVptQkVOdGlHNkZMNHlpMU5MK1BXY0F2Vnc4NmtRQmxr
VHZsU3VNdmFad1Exa1JJcncxZ3UvdUVHY1U1Q09hY1l3dktUdUwvR01oMTBnNWUwR1ljWUVkSEN0alRF
NUNDV0NPWXQvdFRMZ05rZVYyUTdRZFk1WGs5WnNjM3pTMUtyd3I0RGJzOUhrNFdDV0hob000WlVuM
zVGMDhiczJwMk4wYmRlYWcxQlExV24vdEhkSjNsMlpicEFxSkdrWllqNEJjRVcrM29JOVZtYVFPSStlSUdOT
2R2NEl5ZGxPdG96OGJMNTBMcXhhQ01HSUF1NUJDUVVlUlpLT205NVN2Y0xab2NHK1VDUEtucmZWW
GtDUHoyMG5od01YaDZaUEsvdkJDb0VZZjJJQ1NNSmZwdkVRUkZLZU0vajRTWVNqbS9DMEMzUW5lZU
N0T2p4bUNyd3Fpbktxcnprb U1wTlRZWFpqVXVhM1d0R2pxNEg0bnZ3VkhLR1NML1pTczZPaWZCTzNZQk
ZFWWNIcUM1OFp0WUdCYWZGSCthYVRkTlFxTXBkOFgwVTM3cmd0cWZlZzA2TlN0K1FUOUV6ajM1U0
w3eWpzKzQrcnZYMlJweWtGRDNzNFMySSszYVZCeHFZNXZUVGxLUDdpRFpIMUFES2puN3JTeGVpdUp
ZaHhxSm5OelF5TU1zZ0dtWWJXM3JuamxzVHFLVS9YRDZ0d09jRWIxN0tTSUk5KzJKcDNNR2pjTFBkWW
RXS3Npb2JkYW1kVTlDajZtSWhHNWQxRFRoTmFqd2RUWGpWelA3YVBzbWpKdE5hTVBGUnlybDYyd1N
pVUZ0TFNsTEptcEY5Vzk4dW1ReVdwZldYSjl5T3gxekNSNjc0WDRKRWlUNndhUDFjR1Y3WkdrbG1ZYnIv
UTNYU2ZqSkFNS1p2cU9wblZMcHpad09HVnE0b1FXOURSRkVMNnd5M0F4MnlleXd5a2RxVWVBMlRGW
ThlbzQ3WDNXQU85SXlYOVpQQVMydmFKTEIvWEhsYXZPSGMzcDNPV1QzSUdwcTZuZ3NPMjFGb2tWbl
ZEdkkrS2ZHPC94ZW5jOkNpcGhlclZhbHVlPjwveGVuYzpDaXBoZXJEYXRhPjwveGVuYzpFbmNyeXB0ZWR
EYXRhPg==</mod:ApplicationRequest>
            </cor:deleteFilein>
        </soapenv:Body>
</soapenv:Envelope>
```


Signing the SOAP message (transport signature)

The last step is to create the transport signature on the SOAP message. The signature is based on a customer signing certificate, and the style of the signature is detached. This corresponds to step 12 on the list on page 5:


```
<soapenv:Envelope xmlns:mod="http://model.bxd.fi" xmlns:cor="http://bxd.fi/CorporateFileService"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:sign="http://danskebank.dk/AGENA/SEPA/SigningService">
    <soapenv:Header>
        <wsse:Security soapenv:mustUnderstand="1" xmlns:wsse="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
            <wsu:Timestamp xml:id="Timestamp-5293c3c7-b69f-48e2-866e-9fc9cd35a422"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
                <wsu:Created>2021-05-07T11:26:49Z</wsu:Created>
                <wsu:Expires>2021-05-07T11:31:49Z</wsu:Expires>
            </wsu:Timestamp>
            <wsse:BinarySecurityToken xml:id="SecurityToken-192f1eb9-6c10-482c-8719-
76754a128e84" EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-
security-1.0#Base64Binary" ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-
token-profile-
1.0#X509v3">MIIDnzCCAoegAwIBAgIHDSQ4jQY0sTANBgkqhkiG9w0BAQsFADCBwjEQMA4GA1UEAxMH
REJHQ0FEQjELMAkGA1UEBhMCREsxEzARBgNVBAcTCkNvcGVuaGFnZW4xEDAOBgNVBAgTB0Rlbm1
hcmsxGjAYBgNVBAoTEURhbnNrZSBCYW5rIEdyb3VwMRgwFgYDVQQLEw9EYW5za2UgQmFuayBBL1M
xGDAWBgNVBAUTDzYxMTI2MjI4MjIzMDEwMTEJMAcGA1UEBBMAMQkwBwYDVQQqEwAxCTAHBgNVB
AwTADEJMAcGA1UEERMAMB4XDTEwMDUwNzExMDIwM1oXDTEyMDUwNjExMDIwM1owgaYxIDAeBgN
VBAMTF0JFTkdUU1NPTiBPRyBGUi4gSkVOU0VOMQswCQYDVQQGEwJESzEcMBoGA1UEChMTREJUU
yBERU1PIDEuICgzNEFLKTEeMBwGA1UECxMVQ09SUE9SQVRFIERFVkVMT1BNRU5UMTcwNQYDVQQ
FEy5TRS1LRVJIVi9EQUJBOjAwOTIxMDA4NDEtQUdSOjA2MTA0OC1VU1I6MDYxMTMzMIGfMA0GCSqG
SIb3DQEBAQUAA4GNADCBiQKBgQCVi1Bq4B6VNK8x6ISrNQIPBI3izM/RNcyGHAE/9FTdqFjMOhZygT0m
0K5SQjLY0AftdoHmLDtEbnzOzGC1x+y/pM8DbfmmDRXV6oC4jzq97AjFTUYI0tJrAe1Fc10mh4SnstWVYaA
hsZu8LxSPyC4JBB+66msvK4RgkLauTZw3/QIDAQABozgwNjAJBgNVHQ4EAgQAMBkGA1UdIwQSMBCAD
sLGxMP29vnBwsTCxsTDMA4GA1UdDwEB/wQEAwIGwDANBgkqhkiG9w0BAQsFAAOCAQEAREM7TjPXuf
```

```
RxrYM90hCP373oRBwrpFqZaXsEo0BDegbZ5ASks5YXZL6OAjU2Ax93wJwpKn1yDBHCkhdZcC426vhDOQ
mjdSiL6mr6OQ8+YSCmlTcYM7xZxYYHAW0MBOJCfNZ0rx32OoJV9gSENq408pu23j4nPrTpIdXUovLsLBV
aGpeR6mGUnjRYUwFXt3N73kWx+z6U0SFS0rPA2dmzt3WFxzodxdyvj4KWpLEOrsVjAYTMLCNqTycYiUYZ
eDFGDrwyAfVlh4zhCYtrrPTPhDmOokhzQTv9B2W/8tNWNFNudHGHmm4DsJozPuj1aeWe70nTfgojpVC/hl
Wl1KQFbg==</wsse:BinarySecurityToken>
            <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
                <SignedInfo>
                    <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
                    <SignatureMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
                    <Reference URI="#Timestamp-5293c3c7-b69f-48e2-866e-9fc9cd35a422">
                        <Transforms>
                            <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
                        </Transforms>
                        <DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
            <DigestValue>GHMVWKn7azl//fyS0ALv9Rnt/9vZieUb51SgRVDYP14=</DigestValue>
                    </Reference>
                    <Reference URI="#Body-89bd1994-b8b8-4b7b-a58b-593dcfd671a5">
                        <Transforms>
                            <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
                        </Transforms>
                        <DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
            <DigestValue>LRpIGFvrFjWvG7X4oohojCyirQbLPuNWQAC29Uwg1Lk=</DigestValue>
                    </Reference>
                </SignedInfo>

        <SignatureValue>HCRlddnNZTElpF1avxlFu0EkCb7tizEsdCmPFdCH2wFTVW5pW6/5l2aDkDobu5rlHF
r5trH8l42+3sv0lji1DcTEfsMLkdhQY1n0TAFX6LBZSP11inP9ydGoWsOm3pnGwC8nXm4zXlmhJFczCSQRG
70b+pCF2Ogc34Eh3FUT8MA=</SignatureValue>
                <KeyInfo>
                    <wsse:SecurityTokenReference xmlns="">
                        <wsse:Reference URI="#SecurityToken-192f1eb9-6c10-482c-8719-
76754a128e84" ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-
1.0#X509v3"/>
                    </wsse:SecurityTokenReference>
                </KeyInfo>
            </Signature>
        </wsse:Security>
    </soapenv:Header>
    <soapenv:Body xml:id="Body-89bd1994-b8b8-4b7b-a58b-593dcfd671a5">
        <cor:uploadFilein>
            <mod:RequestHeader>
                <mod:SenderId>061133</mod:SenderId>
                <mod:RequestId>33</mod:RequestId>
                <mod:Timestamp>2021-12-17T09:30:47Z</mod:Timestamp>
                <mod:Language>EN</mod:Language>
                <mod:UserAgent>CustomerSoftwareId</mod:UserAgent>
                <mod:ReceiverId>DABAFIHH</mod:ReceiverId>
            </mod:RequestHeader>
    <mod:ApplicationRequest>PHhlbmM6RW5jcnlwdGVkRGF0YSBUeXBlPSJodHRwOi8vd3d3LnczLm9y
Zy8yMDAxLzA0L3htbGVuYyNFbGVtZW50IiB4bWxuczp4ZW5jPSJodHRwOi8vd3d3LnczLm9y
Zy8yMDAxLzA0L3htbGVuYyMiPjx4ZW5jOkVuY3J5cHRpb25NZXRob2QgQWxnb3JpdGhtPSJodHRwOi8vd3d3LnczLm9y
Zy8yMDAxLzA0L3htbGVuYyNhZXMyNTYtY2JjIi8+PGRzaWc6S2V5SW5mbyB4bWxuczpkc2lnPSJodHRwOi
8vd3d3LnczLm9yZy8yMDAwLzA5L3htbGRzaWcjIj48eGVuYzpFbmNyeXB0ZWRLZXk+PHhlbmM6RW5jcnlw
dGlvbk1ldGhvZCBBbGdvcml0aG09Imh0dHA6Ly93d3cudzMub3JnLzIwMDEvMDQveG1sZW5jI3JzYS0xXzU
iLz48ZHNpZzpLZXlJbmZvPjxkc2lnOlg1MDlEYXRhPjxkc2lnOlg1MDlDZXJ0aWZpY2F0ZT5NSUlEeVRDQ0F
yR2dBd0lCQWdJS05EUTBORFF6TURBd01qQU5CZ2txaGtpRzl3MEJBUXNGQURDQnhERVFNQTRHQTF
VRUF4TUhSRUpIVWs5UFZERUxNQWtHQTFVRUJoTUNSRXN4RXpBUkJnTlZCQWNUQ2tOdmNHVnVhR
0ZuWlc0eEVEQU9CZ05WQkFnVElwUmxibTFoY21zeEdqQVlCZ05WQkFvVEVVUmhibk5yWlNCQ1lXNXJJ
```

RWR5YjNWd01Sb3dHQVlEVlFRTEV4RkVZVzV6YTJVZ1FtRnVheUJlY205MWNERVlNQllHQTFVRUJSTV
BOakV4TWpZeU1qZ3hNVE13TURBeE1Ra3dCd1lEVlFRRUV3QXhDVEFlQmdOVkJDb1RBREVKTUFjR0E
xVUVEQk1BTVFrd0J3WURWUVFSRXdBd0hoY05NRGt4TURNd01USXdNREF3V2hjTk1URXhNRE13TVRJ
d01EQXdXakNCeFRFUk1BOEdBMVVFQXhNSVJFSkhRMUpaVUZReEN6QUpCZ05WQkFZVEFrUkxNUk1
3RVFZRFZRUUhFd3BEYjNCbGJtaGhaMlZ1TVJBd0RnWURWUVFJRXdkRVpXNXRZWEpyTVJvd0dBWUR
WUVFLRXhGRVlXNXphMlVnUW1GdWF5QkhjbTkxY0RFYU1CZ0dBMVVFQ3hNUlJHRnVjMnRsSUVKaGJt
c2dSM0p2ZFhBeEdEQVdCZ05WQkFVVER6WXhNVEkyTWpJNE5EUXpNREF3TWpFSk1BY0dBMVVFQkJ
NQU1Ra3dCd1lEVlFRcUV3QXhDVEFlQmdOVkJBd1RBREVKTUFjR0ExVUVFUk1BTUlHZE1BMEdDU3FH
U0liM0RRRUJBUVVBQTRHTEFEQ0Jod0tCZ1FDcGFlMnROYkNtUFJuaTc0dDRwWHB6VXdzVVRSQmVhe
lBwY2haSGpuK0hCcGZRQUg2NmtDek9jeExlT3JMNzFDRlJajZCNFdycVhaeDRTVWxkUXBLdTY3dlNoNV
pzTHl5WFBabmlDbytEdFgyek9ZR0VpZ3pxNmhJeWxSNVhLblhrWmZKU3hvL1RnR1gzcmtMeEY2Y2JhUD
ZteE43TGNDdjlkVWorZVJubmdRSUJBNk5BTUQ0d0VRWURWUjBPQkFvRUNFUkNSME5TV1ZCVU1Ca0d
BMVVkSXdRU01CQ0FEc0xHeE1QMjlzTDI5L1gzOWNYRU1BNEdBMVVkRHdFQi93UUVBd0lFTURBTkJna
3Foa2lHOXcwQkFRc0ZBQU9DQVFFQUQ4K0Jwbk0vSzEwbTVpS080QlRjdkdpNzdvU3Awc3VsNXV0UDZa
RkV2ZXVxamxNeDR4eGgrMVNWUlNNWFF0dGQ0OS8wOEJJVU9hWXJJanZQK2RKVDVKNkQrVWVGUl
NqRTZtZXZqUkUxcXFJcENzSG9DV1ZFOVdraXVGM1FMYW91RDlQSDVGUHNrNTNRVmRkNFlwK3ZJS0
RWWVFicGMxMkE4d0FFdThoa25ncXhOMi9FcGQ4dS8zTFFnc1RtWTZtbjVlazJFRk5JNm8yZFFhNTAzRH
B6am9YR1VSSUZPb0FhV0l3WlVKaGU3bnhHMVhqbjFPbnR0cmVvUDZ0d0NUckc4SW9MZkdLRGtMVlEz
Rys3dkFxWGdCK0p3RkxVb1VMUmF3MGVJU1BnRTduTWt6MGxhMGx1K2FKbGFlUkpacS9GOWVNa0RY
RnFSU01HSEhOVWlqenJRUT09PC9kc2lnOlg1MDlDZXJ0aWZpY2F0ZT48L2RzaWc6WDUwOURhdGE+P
C9kc2lnOktleUluZm8+PHhlbmM6Q2lwaGVyRGF0YT48eGVuYzpDaXBoZXJWYWx1ZT5CYXhDSXUyWEVt
WHJDSEdmQU1NSC8vVGxma1F5UXlzNnlQUmR0UTc4UTQvMjhTQ2k3Slh1S2hxN09EWFh5OWo0bDRV
U0d0Y09TcTVsSU1QUFE3SnVnVkNRaVhaVkRCVEVLNlRqN3VNL0NQUlljb0Fmb3ZyMkw4QTBtelk3MDR0
YzhDMVZIaXRaUDcyZVZrOUs4aFN3ZTRXK25QaS8wUlVSaUxoMHBhcVNQdjg9PC94ZW5jOkNpcGhlclZh
bHVlPjwveGVuYzpDaXBoZXJEYXRhPjwveGVuYzpFbmNyeXB0ZWRLZXk+PC9kc2lnOktleUluZm8+PHhlb
mM6Q2lwaGVyRGF0YT48eGVuYzpDaXBoZXJWYWx1ZT5MY3pPcHlrbHQ2bXV3dGFjSmdyMk9mdlNMczZ
KMXV5bUZaR28xT2x3NExveG9hWjhZdkMvZm0wa04xOEViQUxtZU8xS2p0cTZ2VWp3MEc4SFlqaGlLanR
OUmpJSTJWbkRoTndZMGQ0cWU0QlNONjNlbFFBZzZabTNmd003MjdyVSs1WC9SZy9wSE9qUDZwamF
PenR1STZEY000Y1YzZmtBTnBGaGRFekhQWlQ5ek9ET0F6Y0wyRDdFam5hL0pnS1NybEVOdEZscWd3N
G1BOVB0c3UrQndWQnprQzA2U3N3VXRsTG9ZYnB1TzZZL2RqMjc2S1Era05Nd1EzZ2FKTUVNZG0xcDBz
RStkU0s2V1U0WHhrNTMzZkc1a21wRU9zWEtjbk1BanJlT25KdFVlUXRiamFLM2Vud2xlRjZpNElwQ0lvZmF
qY0QyUFZBVmdlRXpxQ29DRmN1cEczVWlvR0EzNXRBRTZ0emhjRHNzVUNEaWhFV2NUVlNZVjFMMWo
wR2RDYXhHUTBRdHRhZWRYTVU3OUZoUXg3M2NjcFU0WnU0KzliNEVqRTRkanZMUXBJckdVRnh3Z0I4
NDl5Y0JzdmkrSVYvVG9haEZ3QXNwTHk3dlZaTUh6TnRGcm5uVGxUTFZ1ZVRmZmROZXZOTXdaUGtKc
FJwSUE0ZllmcDMwY2pEY2JqRmkvdlRsRmJJR3lzMnRlZHh4M0JPWkVXOThFeHBYTFZoRGd2SDhMNzF
5MDJaMVlBTCtRTDZxTFZrcDhXWnlROS9nSEo1dTVHZ1BiZFdsNWhKb2w3dTVOaHRGdUROUk95aStQT
DhOUXZLeXFaa1RoNVFWYnFBbjVBK2QxMEdqeW4vZ0JHbUpvRkxRbHFqM0pCamJaNktYYU9QSy8xSE
MrMTkrNUZmQ1NmNnB0eG04RjlSMXVNMzhEdHUvSlBVN1RKM3h1a2tiU1pZUGxQbTFKZnlKRGFlSlYrcH
lvMGVuNHNDMGhDcFRleXJzWHpCblRyTG5tOHFnbkkrMktLK2hFY01kQzRxSVdZeGJReEVOa1NCNXlaW
DdITXUrbThndDNzYnZoT2ozOXUyK09HK2FLKzkwa1VuT3ZMYTJqUGtLSWw5SWEzTXRZemdqbFdZSk1
GQzdacWdwUnRNL0dCckFrYjRxU0IzbTdlVlRYS3NwYzBXT3BaWTZHY1Y2WFcxL21zZCt0N3ZXUUlwQitD
cHhZTEJXNVJHazU3TTNzY2gxeGFlbWwxVW4ycVNaQWxlYS85SjY1QlZkMG1SaDR1ZzNpaWRWdEZIW
UFickMzQWNyNEM3WmlraVl2MW5LWElpekwzRWk1cGZPcHo3YWNpTVk2dDBxQ3hjMllQajR4WUtWdU9
FUHgzUFNDaEVOTStjQlpYMWxWYk52cC9ncTJ4YmdDWXZJemtnYmcxQWxVZitCNnhPVlZ6czF3SnVOM
k1ldnJtVDVMblNuMU1HaVBzMFVWTDl2VTJYT043QjBMN090dis4SVBYL0Myb1NTanZCS1h4Ykg3RFFEN
TNmWTV0N2lNcHZydEFhVzl5cC9ySW5BYm9zN3R5SGpBRXAxeEU5bVdlSk96Vjc0VnpydDVlaGs1dU9xY
zk0Nm4rWERuR21RMDV0SUlqSW50QnhlWFVDY0VoNTRhTXB0SjdZNjBCTmlHSU1EUWpVRXVXU0pZZ
EpHeHFvYjA4Z3IrNVB5UjE0RUJFOWN6S2kvZnpsdlR1enp4RDBwdXFZQnZnNlRyeVB0WVJpN2s4Nk42Zl
R0OU4wYUhHeWhZL25OVWcxcndhZ2xwRThhdkFtREZlQjlzazRJNnZUMXc4OEdxeUpCaHRvbGZOVHdn
WDJjVGpjbjJBTUdkRytzWXdnSmF3VkRpU09GUlB1WnVKaXlEUzd6d2J4Wm13WlE3MjhHNHJKOVFEaFc2
U3R5MXcybzVxL2RZdjJiS0hHaDhZb0NMb0c1RFZoTUp2TnJjUk9aczB4RHNKelVRaVhDMWZ4Tm91WlZR
jU5OHc2N2dMWnZTNFh0Y1VVZUl2UkNoMTNONjZsaFZvUEdZZHFFSEJ2V3lpaVZYbHVjSmJIYkJKOGtza
nhMM242WTVtcjAxRDJLNXFKQW9RMDI3QVRFL282bXY3U3BCQ2EyOE9QMGllajg3b283SnBjVmdrSDVi
ZG9XOHBsNEVYY1BnbVZkTzcwTWgzTjAzSHNCc1NDY0dRWGFnQThmK0hod3JucnZWTENRREVaNStB
czBsbDRFVFRvTi9uSGhHMkRUYUI4ZkZWWk40YU02eW95SmVub0NmNk4vSXZsSi9pWHo1NUtXMkFoZ
3JMNk9TTXFuZkZoRVVSclVJdjJBcC9nanQyYjVtVEh2KytnTTdXQWFwRTB6QkR0R3NQYmFrZWN0VjNEV
kpjZzFFZmFvNEw4WGV3N3c3L0U3MXJGOE4xTzFxd3ZLVzIwZ1J6TDZhekpaajhqRFZabW5lWTNnc1lQck
Q1Y3BReHZFMS82dDczdVQ0UHVkRDhBQVF1V05hMlpaZ0QxbTZyMFZlNEJDbXZERWVKZGNLcmRNY

mdsdDR1RlVGWFl2OXgwWWY4STZSb3N5ekpYcUpUQ21Ba0V3UUFwUEJ3bE50LzRtaVBlaVNlSldWMjg0
cUl4L3gzMTJVVnR1YTdjNm1GZEVMUzlSazlyM0hSVTR6cXM3Z2RDczlUM3A5cDJpNGRINDIrczJBalZyNk
VDWUVsOVZETTI1REZzaHI4WnJEVUNCWlBGcjFLbnhiNDBwR0I0d2JoeTRvNE5mVnIzZlhTMlJlRXNrSXA
2SlloYUFLNUFHR01xblNyUVQ4dWdPbDRrcldXbWJvL0ZiQldxMmtuQVM5LzdicE9ZSE9XVHEyM2RLT3dW
T000TVZKVGlxckRITXNmOUhOdlp6dkNwckxJRGRFTWRFYUZXSWgxbGdQMk8yQkd6ZUR0cFdhR0cxbH
h3N0g5aGdaR2sxbENnSFhSWHd2R25LVUZLUWhCblZxaXk5VEp5M1dZbmYxOGJQRnlVblE3QjZuditpRjJ
EMFprZGExMEhGd0kyMFpnWWIzcFpOakxkelVHOXdKT28xaWo1WkVDUlpDZTh2TzB6R2xZelJTTC9iTElY
UFZjV3FaS1lnbHBMVzd5R0JkdDVRR0t6TlBpM1d3MjNkZEVDcWFyKzZ4SlFYZG1RRk9EWmd6Nk9ZMnc5
bEpaOXhVdG9nMHFNVlNCM1lzREZsOXdNNHovSGJLUTFnK0VKRW5YV0J0V0ZWbWlnenB6VjZuR0RSb
UViNlJQRzF2bmFabmRYbnJZci9yc1JDcU0wVzB2Q2hrRjdDbXpwWWtDUHFWRWNLdnEvc2tCV0lUT1lFd
ENUQmZzbXJaQUtodDdCckdSZVNaalp6U0ZoSXNKM212UnV6MFJaakhPSnJFTDJMYlgzT1R1YmZML0J
BSXo2Z2xLdWZTLzRmdnR0cnpPZzVBV0FROHVlRi9KaUprYXNVcWFnalhtZzVnakFkOHFTQmtmMFY5cjl3
cVFmM1poZEhXZ2xzNXlJZE1yTm5oN2ZRc2hzbGhaYzVYTTd4QVlkRHVuOVoxMkRvK3pJY3pEeDFiOThE
V0dTVkpnaDVGVUx4emwxalhhcEJkRFlCSzJ3WXBiMnVNcTVCQXovbk5SN0Nka3RFdGVTTE4xWGFOL3
ZSYnp5TlU1ckthRDV3VmFrQ0RiYmJvSXBZOUFEOUlvQXlZajAwOEthZTJ6SXNCcW5KNGhSVTllS0ZzWU
JWY0g5MStSZkZtVS9Cb0hZZkw2dUdCTHhHU29FK3J5QlVOMVlCWDlVQkg0R2FHc2lGUTlFcGk0cFJQUF
dkNjJ6SGhxRzd1QUh1bGFkTGNVSjZnWU9Yb292WlQ5S2xPa2ZEZW1QdkpZeU9GWHp5K3VPejMwSGgv
YkVGNjNna25JMUZWY1kwZ0pyRnl3MUFTU2JkVWwrSDBZUHdYNnY0K1pOMjd1S3lyV0dSSGVqUlBrTUt
OTnArVmRuTy9YQWdlbjZFRnd1N0xUMnpVd2JONEt2amVLYTlRM0s1ZWtoVmRyQXUvTHhqSWFHVGJm
S0RqMGg2cDNlSFNvWHFpdE5BaUNXcFRTQXR2Q3BiNnJ5UmxkWkxvOG1sMTgvQkpYaWJDZU5DY3hE
SjFjNEpMemtleWJoR1IzMUFnQTZEZUhuLzArYzFMNS9sZ3RhRHNnbjRpV21TcTZnSzBCUmlpNWZsWHF
WcGxUTVNVU0RhNDNVMmJEdkZiand2OGJCWk1yWjBwUkJ0Z2QwT1ZPdzFObnIvV1dUS3BsTzl0d1NaR
G1SKzBZOXhjNGVBejk0ZWhuTHd2SHU4UHFJb3Y1NHc3cnNUM3JJWXNRU052WVBiWDl5Q1liRFNKbTF
USUx2Yk9sS3RFa3dLNVZzYXdha3NPRml3clcwVEpqa282MFB1YlNQRWc2alZZaElwNkNidXpEN1I3cm9Y
WitVOC83MDZyczF2QnY5ZVdsdjhzK2JUdUtRVW1GUWkwZTNoOHdUVHVsUzFIRXd5eGt1amE1aUd2RT
RNZTcraEdaSlh0elFRZ3lCb3hsZFhHbVJ5RmVld29mOHk2VlRWSXRPU0dBSldJVmx6WnRXM0Uyajc2WC
svNUlVWG95Zmt4bXZvWFdmdTBQOEwrazMwYkRGNlcweC84MEIzQ3Y1dUN4SnA3NlJnQUhkRWZtZ2R
UL0RqTnpvTzR5MTluNlJIUi8vYVJ4ZEVFdHpYbDNybDE5YWJXdXFvZjNjTkwxV0pvT1ZqZEZNdU83N3pJ
WHNUdVVmSm1ZenAxditNNFhCcko4aGxwek9kTzFtMVhVM0JtWEo4T1krOGtscFhyT0FyRS9mSkFjWjdHT
UVJdVBmSFBVM3FOMEdpSTFaUWdDZG1vSDdiclhSL0NxOWhFdW04ZXVjYVlscW95UUhvQ2hValJ5aldh
NmVaOG1VUzY2ZmRrbWdyWTh5UDhNWlEyME1UUGdFdkdaalkwVTJKdWx3bTZ4Q1crSGs3VnNsbExsN
kJIQ2pISHpIRC9sOUlidmIxYy9McnNRMWgxUEdyRGhvZWllSnFpbnB1R0MvM3d3akR6dnFBRncvRGM5M
S81ajdEVitYdEhtaGxvV3V3VWs5UmM5Z05pOTAzSHRiVERud2hETWprVVFaZmw0cXBmVy80OUM2VTV
CaWtSQytCWVlIVjVHR296Z21MMjBTMHNXeGJVYUV1SmZTN0pZNGw3WHRFTjVacUxYcVYxNWZ2QXN
VQ2Z1VVYrS0lFZ3VRd28wbnZEU25kWVAyRE1XQXVlZGpqaklhQ3BKSDJpZlBFdnNXMWlpbldsSlJzNmhG
azVxSVUxcFE0dkJUT2d3S09BOWZLcVl1TVRCL2E0YWErQ0pub1ZMZVcya3dzTkk2MU5GZUJvQzMrUGF
4ZTY3b0lRUXBVSjJBRG96ejRmcEZ6cE8yUnU3ZkxvNGpSVlE9PTwveGVuYzpDaXBoZXJWYWx1ZT48L3h
lbmM6Q2lwaGVyRGF0YT48L3hlbmM6RW5jcnlwdGVkRGF0YT4=</mod:ApplicationRequest>
            </cor:uploadFilein>
        </soapenv:Body>
</soapenv:Envelope>

The XML examples are distributed along with this document.

Appendix B: Web service URL

The URL of the web service is:

https://businessws.danskebank.com/financialservice/edifileservice.asmx

It is not possible to fetch the web service wsdl by appending '?wsdl' to the URL. Instead, the wsdl must be retrieved from the bank homepage.