PKI Web Services

Danske Bank

Application Security Infrastructure

This document describes PKI Web Services implemented by Danske Bank. The PKI Web Services enable bank customers to get and manage certificates for use in secure communication with the bank.

1. Table of contents

2. Log of changes

Version	Author	Date	Change
0.1	Mikkel T. Jensen	21.09.2009	Document created
1.0	Mikkel T. Jensen	23.09.2009	Revised after review.
1.1	Mikkel T. Jensen	10.12.2009	KeyGeneratorWDetails element removed from CreateCertificate and RenewCertificate. SSL encryption of web services added. HMAC signature on CreateCertificateRequest replaced by PIN.
1.2	Mikkel T. Jensen	16.04.2010	Revocation changed to revoke both signing and encryption cert. RequestId element moved inside actual request elements (schema change). WSDL changed to only require encryption of CreateCertificate and RenewCertificate requests. Reference to 'Encryption, Signing and Compression in Financial WS' added.
1.3	Mikkel T. Jensen	26.04.2010	Updated example XML.
1.4	Mikkel T. Jensen	27.05.2010	CRL distribution added. Id attribute on request and response elements changed to xml:id (schema change). Various metadata elements added to headers and request elements (schema changes).
1.5	Mikkel T. Jensen		Additional error codes added.
1.6	Michael Andersen & Lea Troels Møller Pedersen	05.11.2010	Bank CA certificate moved from GetBankCertificates to CreateCertificate and RenewCertificate. Field "CAID" deleted from input to all operations. CreateCertificate: - Added CACert as output - Changed returncodes, so no extra information is given about whether a user exists. RenewCertificate: - Added CACert as output - Changed returncodes so general codes are used - Timestamp removed from output RevokeCertificate - CRLReasoncode changed from string to int. CertificateStatus: - Removed status "unknown" (now an error). - CRLReasoncode changed from string to int. - Added CertificateType to output to indicate signing or encryption cert - Added MatchingCertificateSerialNo to output to identify matching signing/encryption certificate. GetOwnCertificateList - Added CertificateStatus to output in order to save an extra call to CertificateStatus operation GetBankCertificates: - Removed input field KeyGeneratorType - Removed input field CustomerId - Removed output field BankCACert Added description of format for CertificateSerialNo. Appendices now only represented by a letter. Illustrations of XML structure now before field explanations. Error codes grouped and updated.
1.7	Lea Troels Møller Pedersen	25.11.2010	Updated description of the customer test environment.
1.8	Lea Troels Møller Pedersen	08.12.2010	Added UTC timestamp requirement.

1.9	Michael Andersen	14.12.2011	Added appendix D about a problem with verification of XML-DSIG signatures in .NET.
2.0	Thomas Malmstrøm	10.07.2012	Changed GetOwnCertificateList to include revoked and expired certificates.
2.1	Lea Troels Møller Pedersen	15.11.2012	Name change to Danske Bank.
2.2	Prashanth Rai	22.02.2016	6.5 Updated the Environment attribute should be "PRODUCTION" instead of "production"
2.3	Mikkel T. Jensen	05.12.2016	Clarified that error responses are not signed. Wsdl and xsd updated to have local schema references for XMLDSIG and XMLENC schemas. Type added to ExceptCertificateSerialNo element in xsd. Undoing faulty update regarding Environment attribute made in version 2.2.
2.4	Andreja Andric	27.07.2017	Reformulations of several phrasings according to suggestions from internal security experts
2.5	Mikkel T. Jensen	18.09.2018	Updated section on certificate distribution and renewal.
2.6.	Andreja Andric	21.03.2023	Changed the title to PKI Web Services, fixed broken links and removed the obsolete Appendix D.
2.7.	Andreja Andric	18.08.2023	Changed the Environment values to the correct value *customertest*, wherever it indicated customer test environment, but was stated in some other way, like *test*, *systemtest* and the like. Added that RequestId has a length limitation of up to 10 characters.
2.8.	Andreja Andric	05.09.2023	Updated the algorithms in examples.
2.9.	Andreja Andric	01.08.2025	Added clarification of SOAP version (1.1)

3. Introduction

This document describes the PKI Web Services implemented by Danske Bank. The services are to be used by customers for certificate creation and management. The certificates are to be used for communication with the bank, e.g. to communicate with EDI Web Services.

4. Channels of communication

The services described in this document will be available as SOAP web services. Other channels may be added later. The web services will be protected by SSL encryption, as well as encryption applied at the message level.

The web services to be implemented are further described in WSDL file PKIService.wsdl and in the XML schema file PKIFactory.xsd. The URL of PKI service is:

- https://businessws.danskebank.com/ra/pkiservice.asmx

5. Standards used

The PKI service is based on the following standards:

- XMLDSIG: *XML Signature Syntax and Processing (Second Edition)*. http://www.w3.org/TR/xmldsig-core1/
- XML Encryption: *XML Encryption Syntax and Processing*. http://www.w3.org/TR/xmlenc-core/
- X.509v3: *Internet X.509 Public Key Infrastructure Certificate and CRL Profile*. http://tools.ietf.org/html/rfc5280
- PKCS#10: *PKCS #10: Certification Request Syntax Specification Version 1.7*. http://tools.ietf.org/html/rfc2986
- SOAP: *Simple Object Access Protocol.* We are supporting SOAP version 1.1. http://www.w3.org/TR/soap/
- WSDL: *Web Services Description Language (WSDL) 1.1*. http://www.w3.org/TR/wsdl

6. About the PKI service

The security used by the bank is based on PKI (public key infrastructure), which is based on public-private key cryptography. In the bank setup, every customer has two certificates: one for signing, and another for encryption. When the customer sends files to the bank, customer signs the message using customer's private key corresponding to the customer's signing certificate. When the bank sends files to the customer, bank encrypts the message using customer's public key corresponding to the customer's encryption certificate. The customer decrypts the message using customer's private key corresponding to the customer's encryption certificate. Similarly, customer encrypts message to the bank using bank's public key corresponding to the bank's encryption certificate, while bank signs message to the customer using bank's private key corresponding to the bank's signing certificate. Table 1 shows this relationship.

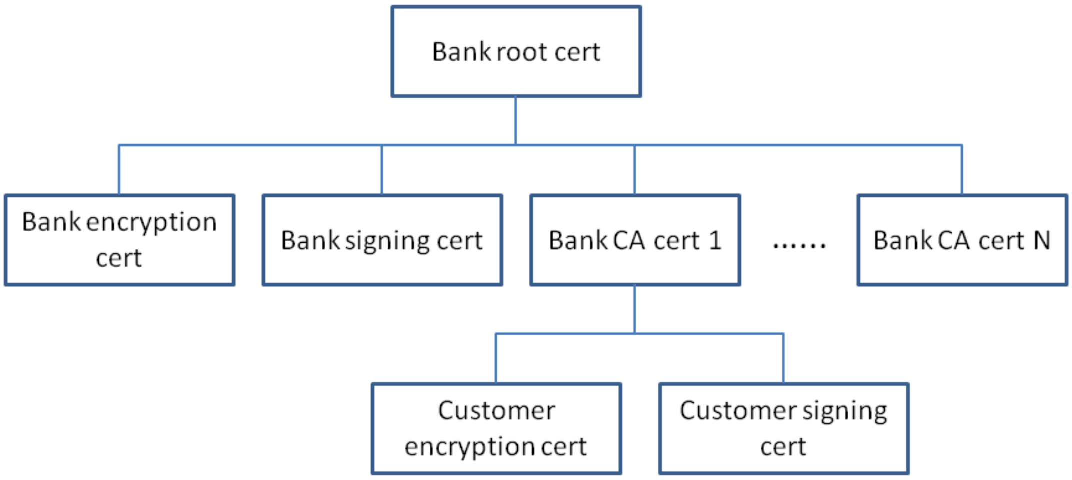
	Encryption	Decryption	Signing	Verification
Customer to Bank	Bank's encryption certificate	Bank's private Key	Customer's private key	Customer's public signing certificate
Bank to Customer	Customer's encryption certificate	Customer's private Key	Bank's private key	Bank's public signing certificate

Table 1. Summary of PKI service

The certificates are organized in a hierarchy, the root of which is the bank *root certificate*. The bank root certificate is a self-signed certificate and it is used to sign the other bank certificates. The bank certificates are:

- The root certificate. This certificate is self-signed, and is obtained by the customer through download from the bank homepage. The certificate is obtained in a package signed by VeriSign, and the customer should verify this signature before trusting the certificate received in the package. After this, the bank root certificate can be used by the customer software to verify the other bank certificates.
- The CA (Certificate Authority) certificate. This certificate is used to sign customer certificates. The bank has several CA certificates, the one relevant for a specific customer depends on the customers relationship to the bank as well as the mechanism used by the customer to generate the keys (generated in software or by a hardware security module).
- The encryption certificate. This certificate is used to encrypt messages sent to the bank.
- The signing certificate. This certificate is used to sign messages sent from the bank.

The hierarchy of certificates is displayed in the diagram below.



6.1. Certificate distribution and renewal

Every certificate in the hierarchy has a certain lifetime. The lifetime of the bank root certificate is 20 years. For the other certificates, the lifetime is shorter (two years or less). Every time a certificate is about to expire, a new certificate must be issued and distributed.
Regarding the customer certificates, the customer must call the RenewCertificate operation in order to get a new set of certificates. This must be done before the old certificates expire.
Regarding the bank certificates, initial distribution of the root certificate, when the customer has never used the PKI WS system before, is handled through another channel than the PKI service. Contact support for details.
Once the customer has a valid root certificate, he uses the GetBankCertificates operation to fetch the encryption and signing bank certificates. The operation is also used to fetch a new bank root certificate if it is available. This is possible since a new root certificate is issued well before the old root certificate expires, which means that the old root certificate and its associated signing certificate can be used to verify the response in which the new root certificate is received.
The intermediate CA certificate used for a particular customer certificate is distributed along with the customer certificates in the CreateCertificate and RenewCertificate operations. The lifetimes of the intermediate CA certificate follow the lifetime of the issuing root certificate.

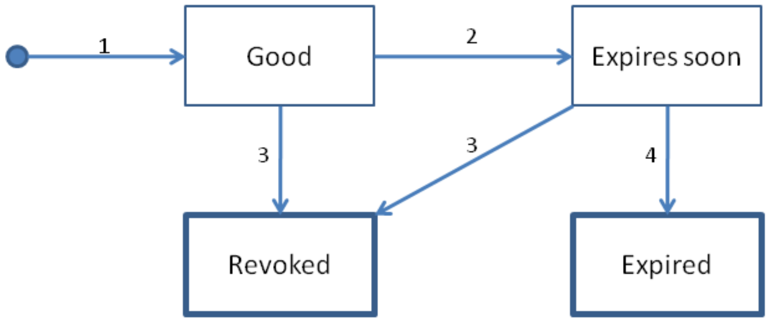
Regarding the RSA key sizes of the certificates, the root and intermediate CA certificates are based on 4096 bit RSA, while all other certificates are based on 2048 bit RSA.

6.2. Certificate status

Each certificate has a status. The status of a certificate can be any of the following:

- Good: The certificate is valid, and will continue to be valid for a reasonable period of time.
- Expires soon: The certificate will expire soon, and should be renewed.
- Expired: The certificate has expired and is no longer valid.
- Revoked: The certificate has been revoked.

The following transition diagram shows the possible transitions between the statuses:



The bold boxes are final states. The statuses are changed by the following incidents:

Incident number	Description
1	The certificate is issued. Prior to this, the certificate does not exist.
2	Time goes by, and the certificate gets sufficiently close to the expiry date.
3	The certificate is revoked.
4	Times goes by, and the certificate expires.

6.3. The certificate generation process

When a customer needs to create an encryption and signing certificate pair, he generates two public/private key pairs on his computer. The public keys are placed in PKCS#10 requests, which are sent to the PKI service, using the CreateCertificate or RenewCertificate operations. The bank inspects the PKCS#10 requests, and if all goes well, it issues certificates, signs them with the CA certificate and sends the certificates to the customer. Notice that the private keys of the customer are never revealed to the bank or to anybody else. Thus, since only the customer has access to the private keys, only the customer is able to sign documents with the customer signature, and only the customer is able to decrypt messages encrypted with the customer encryption certificate.

The bank strongly recommends that customers generate and store the keys using a HSM instead of software. This is because the private keys are much more difficult to steal if they are stored in a HSM. The CreateCertificate and RenewCertificate operations take as input the source of the keys (software or HSM). It is up to the customer to set this parameter correctly, since there is no way for the bank to verify the source of a specific key. The reason the bank needs to know the source of the key is that certificates based on HSM keys may be granted a longer duration, and that in certain situations in which bank security is attacked, it can be advantageous to know the source of the keys.

The customer's public key has to be 2048 bits long (shorter keys will be rejected). The length of the bank's public key is also 2048 bits.

6.4. Bank CRL distribution

The bank distributes a CRL (Certificate Revocation List) of revoked bank certificates. Only bank certificates will be in the list, while revoked customer certificates will not be found in the CRL. The CRL is returned as a standard ASN.1 DER encoded CRL according to RFC3280. The CRL is issued every 24 hours, and is valid for 48 hours. It is the responsibility of the customer software to regularly fetch the updated CRL, and to validate the bank certificates used in communication against the CRL.

The CRL will be distributed via http at the endpoint specified in the root certificate.

6.5. Calling the service during development

During integration to the service, it is possible to call some of the operations in a *test-mode*, which makes sure no changes are made to the bank database. While working in test-mode, the operations effectively work in a dummy mode, which means no real certificates are issued, and no revocations are made. This also means that the return values from operations working in test-mode will be dummy data.

Test-mode is available for all operations but GetBankCertificate behaves the same in test-mode and production mode. CertificateStatus and GetOwnCertificateList are safe to call in production mode in the sense that they do not make any changes to the bank databases. However, test-mode is provided so it is easier to test that different output is handled correctly.

The operations are called in test-mode by setting the Environment attribute found in the RequestHeader to 'customertest'. For CreateCertificate, RenewCertificate, and RevokeCertificate the Environment attribute must also be set to 'customertest' in the operation specific request element.

When calling the real operations, the Environment attribute must be set to 'production' or left out completely, which defaults to 'production'.

Operation	**Behavior in test-mode**
All	The following is checked for all operations before anything else: • Compliance with the WSDL and XML schema • Correct signature (when applicable) • Understandable encryption (when applicable) • Uniqueness of RequestId, CustomerId, and Timestamp combined • Timestamp is neither too old nor too far in the future

CreateCertificate	The pin 1234 will give a valid response back. Anything else gives return code 20.
RenewCertificate	Any PKCS#10 request that starts with “M” gives a valid response. Any other value in the PKCS#10 request gives return code 10 suggesting a problem with the PKCS#10 requests.
RevokeCertificate	The serial numbers 5169000000001702 and 3379000000001502 will not be found. Anything else gives a valid response back.
CertificateStatus	Any serial number not ending in 01 or 02 will not be found. For other serial numbers the following statuses are reported: • Serial number starting with 0: Status “Revoked”. Expiry date and revocation time are hardcoded. • Serial number starting with 1: Status “Expired”. Expiry date will be yesterday. • Serial number starting with 2: Status “Expires soon”. Expiry date will be tomorrow. • Any other serial number: Status “Good”. Expiry date will be a year from current date.
GetOwnCertificateList	4 hardcoded certificates (2 certificate pairs) are returned. 2 will have status “Good” and 2 will have status “Expires soon”. Expiry dates are are formatted identical to those from CertificateStatus. CustomerId “207161” or RequestId “123456789” gives return code 11, “User not authorized”.
GetBankCertificate	Same behavior as production mode.

6.6. The RequestId element

Each request element has a sub-element called RequestId. RequestId should be 10 characters or less in length. In combination with the CustomerId sub-element and the Timestamp sub-element, this ID should always be unique.

6.7. The Timestamp element

Every request has a sub-element that is named Timestamp. This element contains the time at which the request was generated. UTC (Zulu) time must be used. Requests with timestamps that differ more than 10 minutes from the current time will be rejected.

6.8. Regarding digital signatures and encryption

The specifications for XML encryption and XML digital signatures contain some degrees of freedom as to how the elements inside the encrypted data and digital signatures should be used. The encrypted data and digital signatures used for the PKI service should satisfy the requirements described in the document

Encryption, Signing and Compression in Financial Web Services

which can be obtained from the bank.

Example XML satisfying the requirements can be found in appendix B.

6.8.1. References and id-attributes in signatures

The signature type used in requests and responses is *enveloped style.* For requests, the signature must sign the actual request element (e.g. the RenewCertificate element). The remainder of the request is not signed. In responses, the signature will cover the actual response element (e.g. RenewCertificateResponse). The remainder of the response is not signed.

In responses from the bank, the signatures will reference the xml:id attribute of the response element. An example of this is the response found in appendix B. The RenewCertificateResponse element has the attribute xml:id="response", while the enveloped signature contains a Reference element with an attribute URI="#response". This means that the signature on the response element can be verified while the response element is inside the SOAP message, as well as in a context where the surrounding XML has been stripped.

Customers are encouraged to use the same kind of referencing when signing requests, but since signatures on requests will be verified in a context in which the SOAP message has been stripped and in which the Request element is the root element, signatures containing Reference elements with URI="" (whole document) references will also be accepted, provided the signature covers the actual request element and none of its sibling or parent elements.

7. Operation and message types

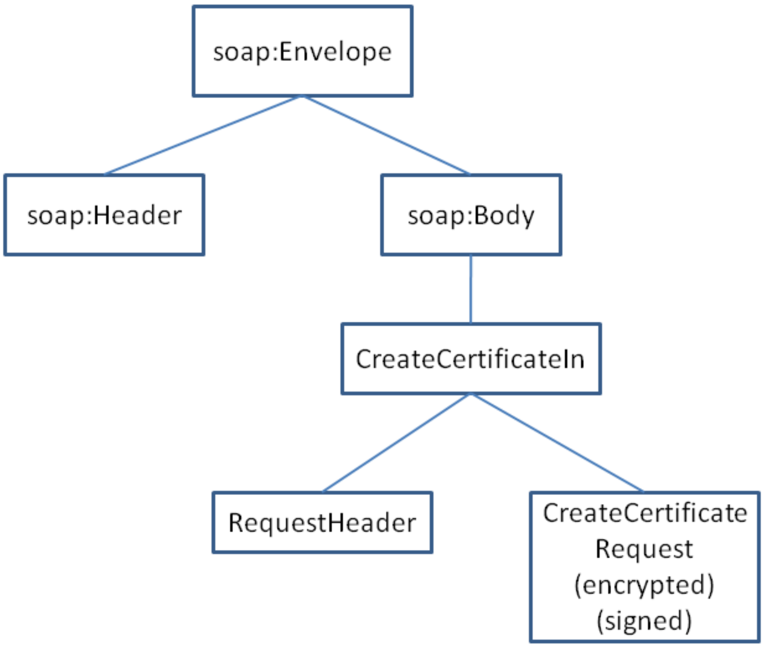
The operations and messages are described in a WSDL file (PKIService.wsdl) and an XML schema file (PKIFactory.xsd). The operations are the following:

- CreateCertificate
- RenewCertificate
- RevokeCertificate
- CertificateStatus
- GetOwnCertificateList
- GetBankCertificates

For each operation there is a Request element (e.g. CreateCertificateRequest) and a Response element (e.g. CreateCertificateResponse). The operations share header elements in the form of the RequestHeader and ResponseHeader elements.
In the next sections, the general aspects of the operations as well as the specific requests and response will be described.

7.1. Generally about all of the operations

All of the operations use the following message structure:



The diagram is based on the CreateCertificate operation, but the structure is the same for all of the operations. Since the transport used is SOAP webservices, the topmost element is a SOAP envelope. This contains a SOAP header and a SOAP body. The soap body contains the CreateCertificateRequest and a RequestHeader. The header contains information such as request id and environment identification (customertest, production), while the actual request data are located in the CreateCertificateRequest element. This element is signed by the sender, and encrypted while in transit.

The structure is the same for all of the requests, except the CreateCertificateIn and CreateCertificateRequest elements are replaced by other elements (the names match "*In" and "*Request"). The same structure is used for the responses. For these, the elements named "*In" are replaced by "*Out", while the request elements "*Request" are replaced by "*Response" elements. For responses, the RequestHeader is replaced by a ResponseHeader.

Regarding the use of encryption, messages containing sensitive information are encrypted, while messages without such content are not encrypted. In this context, sensitive information is requests for certificates (since they may be based on a secret PIN, and since they may contain a challenge password to be used for revocation at a later time[1]). When encryption is used, the relevant request element is replaced by an EncryptedData element from the XML encryption standard. This element can be decrypted into the request element using standard tools.

Regarding signatures, most of the messages are signed, since the signatures are necessary to guarantee that the other party is in fact who he claims to be, and in order to ensure the messages are not changed in transit. The only request elements which are not signed are the GetBankCertificateRequest and the CreateCertificateRequest.

[1] The challenge password functionality for revocation will not be supported in the first version of the PKIService, but it may be added at a later stage.

- Regarding GetBankCertificateRequest, the element is not signed since the operation must be callable in a situation where the client does not yet have a certificate. Since there is nothing sensitive about the bank certificates, there is no risk in not signing these request elements.
- Regarding CreateCertificateRequest, it is impossible for the client to sign the request using a certificate (since the client has no certificate when calling the operation). Instead, authentication is done using a PIN inside the request. The integrity of the request is secured by encryption.

All of the response elements are signed, except error-responses. The signatures used are enveloped-style.

All of the operations return ReturnCode and ReturnText elements. If the operation failed, they will contain information about the error (the elements will be contained in a PKIFactoryServiceFault element, and there will be no response element). If the operation succeeded, they will be contained in the response element. PKIFactoryServiceFault elements are neither signed nor encrypted.

The operations that have “CertificateSerialNo” as input or output expect the content of the field “Serial Number” from an X.509 certificate. The format must be decimal (e.g. 6230000005018301) and not hexadecimal (e.g. 162226E93FF2BD).

7.1.1. Header content

The RequestHeader element contains the following sub elements:

Sub element	Meaning
SenderId	String identifying the sender of the request. In some cases, the SenderId will be identical to the CustomerId, in other cases it may be something else. The customer will be told what value to use.
CustomerId	String identifying the customer of the request. The content of the element must match the content of the CustomerId element found in the actual request element. The customer will be told what value to use.
RequestId	String identifying the request. The combination of CustomerId and RequestId must be unique for 3 months. The content of the element must match the content of the RequestId element found in the actual request element.
Timestamp	The time at which the request was sent. UTC (Zulu) time must be used.
InterfaceVersion	A string identifying the version of the PKIService. The current version is 1.
Environment	A string identifying the environment. The environments relevant for customers are: • production: The production environment issuing certificates to be used with the banking software. • customertest. A test environment for customers to use when they are creating the integration to the PKIService.

The ResponseHeader element contains the following sub elements:

Sub element	Meaning
SenderId	The value from the RequestHeader is echoed.
CustomerId	The value from the RequestHeader is echoed.
RequestId	The value from the RequestHeader is echoed.
Timestamp	The time at which the response was generated. UTC (Zulu) time will be used.
InterfaceVersion	A string identifying the version of the PKI service. The current version is 1.
Environment	The value from the RequestHeader is echoed.

7.1.2. Faults

If an error occurs during the processing of an operation, a PKIFactoryServiceFault is returned. If InterfaceVersion is 1, the PKIFactoryServiceFault is wrapped in a normal SOAP envelope. PKIFactoryServiceFault is not signed. The fault contains the following sub elements:

Sub element	Meaning
SenderId	The value from the RequestHeader will be echoed, if possible.
CustomerId	The value from the RequestHeader will be echoed, if possible.
RequestId	The value from the RequestHeader will be echoed, if possible.
Timestamp	The time at which the response was generated. UTC (Zulu) time will be used.
InterfaceVersion	A string identifying the version of the PKI service.
Environment	The value from the RequestHeader will be echoed, if possible.
ReturnCode	A code identifying the type of error.
ReturnText	A textual description of the error.
AdditionalReturnText	In some cases this element will contain extra information regarding the error. This element is optional.

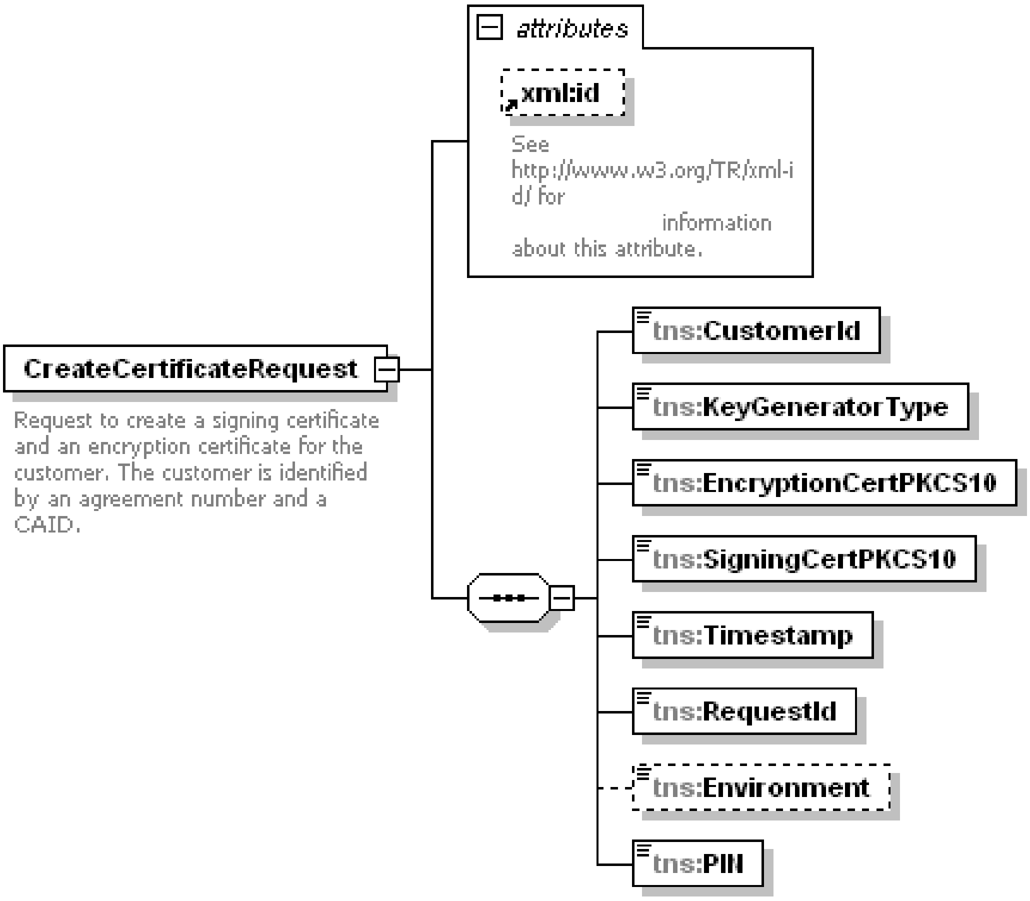
A list of error codes can be found in appendix A.

7.2. Specific operations

In this section, the different operations, their parameters and return values are described.

7.2.1. CreateCertificate

The CreateCertficate operation is used if the customer has no valid signing certificate (e.g. the first time the customer needs to create certificates). The parameters in the CreateCertificateRequest are as follows:

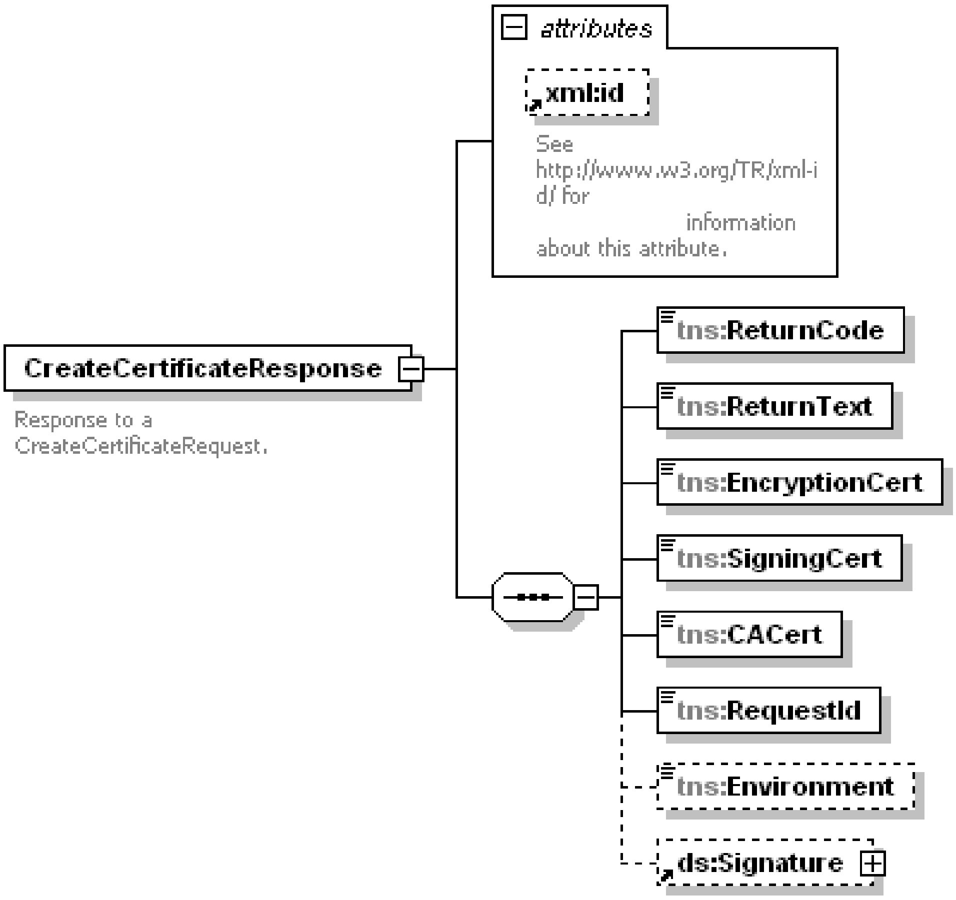


Sub element	Meaning
CustomerId	A key identifying the customer. Each customer will be informed about the value to use.
KeyGeneratorType	Information about the mechanism used to generate the keys. The element KeyGeneratorType should hold the value 'HSM' if the keys are generated by and stored in hardware and 'software' otherwise.
EncryptionCertPKCS10	A base64binary encoded PKCS#10 request. The request contains the public key of the encryption certificate to be issued. The PKCS#10 specification allows for other attributes, but these are not used in the resulting certificate. The bank's existing information about the customer will always be used.
SigningCertPKCS10	A base64binary encoded PKCS#10 request. The request contains the public key of the signing certificate to be issued, as well as other attributes (e.g. name). The PKCS#10 specification allows for other attributes, but these are not used in the resulting certificate. The bank's existing information about the customer will always be used.
Timestamp	The time at which the request was generated. UTC (Zulu) time must be used.
RequestId	String identifying the request. The combination of CustomerId and RequestId must be unique for 3 months.

Environment	A string identifying the environment. The environments relevant for customers are: • production: The production environment issuing certificates to be used with the banking software. • customertest. A test environment for customers to use when they are creating the integration to the PKI service. If the element is not present, the production environment will be used.
PIN	The PIN code.

All of these parameters except Environment are mandatory.

The return values of the operation are contained in the CreateCertificateResponse element:

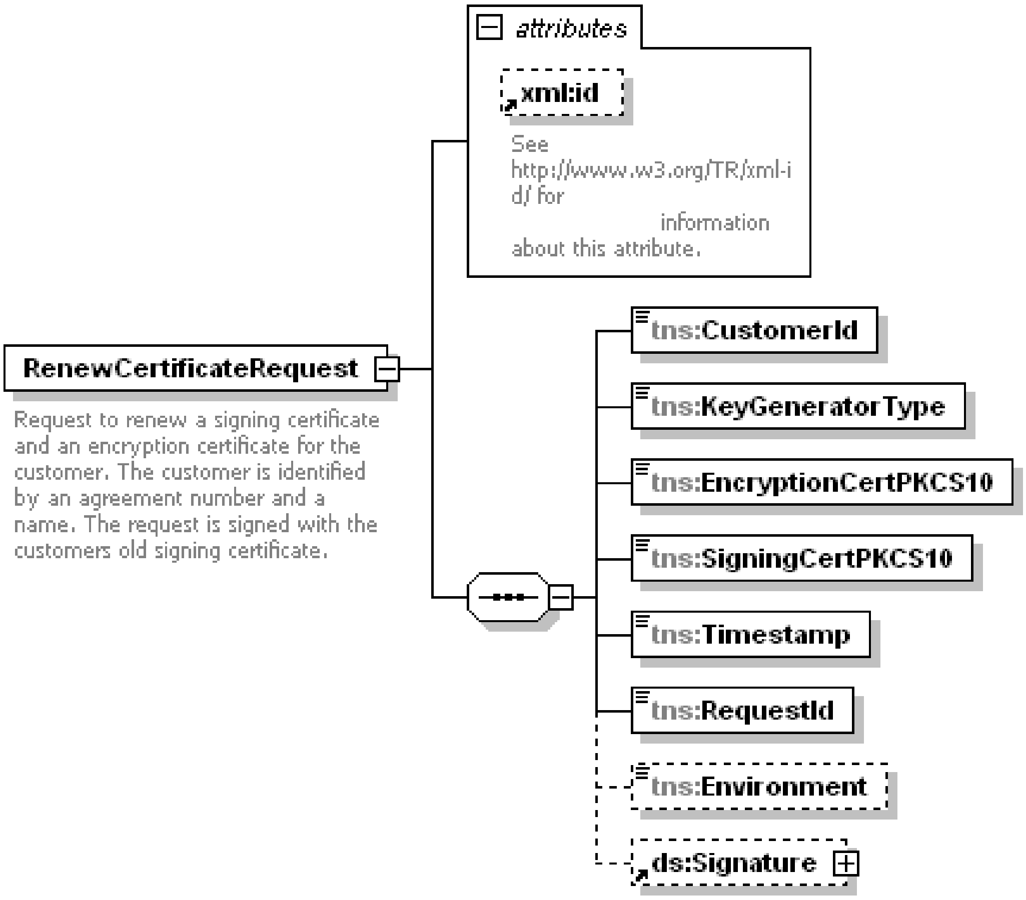


Sub element	Meaning
ReturnCode	The return code of the operation call. If the code is '00', all went well. If the code is '20' then any of the following could have occurred: - CustomerId does not exist - CustomerId is locked - wrong PIN - PIN has already been used

	- PIN has expired - there is no PIN assigned to the user The reason that these are all covered by 1 code is to avoid letting out information about whether or not a particular CustomerId exists in the system (prevent systematic exploits). Other possible return codes are of a more technical nature concerning SOAP, encryption etc. and are listed in appendix A.
ReturnText	A text describing the status of the method call.
EncryptionCert	The base64 encoded customer encryption certificate.
SigningCert	The base64 encoded customer signing certificate.
CACert	CA certificate that EncryptionCert and SigningCert are issued under. This will always be issued under the current root certificate, which is distributed by the operation GetBankCertificates.
RequestId	String identifying the request. The RequestId from the request is returned.
Environment	A string identifying the environment. The string sent in the request is echoed back to the customer.
Signature	An enveloped signature signing the CreateCertificateResponse element. The signature is based on the bank signing certificate. The customer should always verify the signature. If the signature does not validate, the result returned should not be trusted, and the bank should be contacted.

7.2.2. RenewCertificate

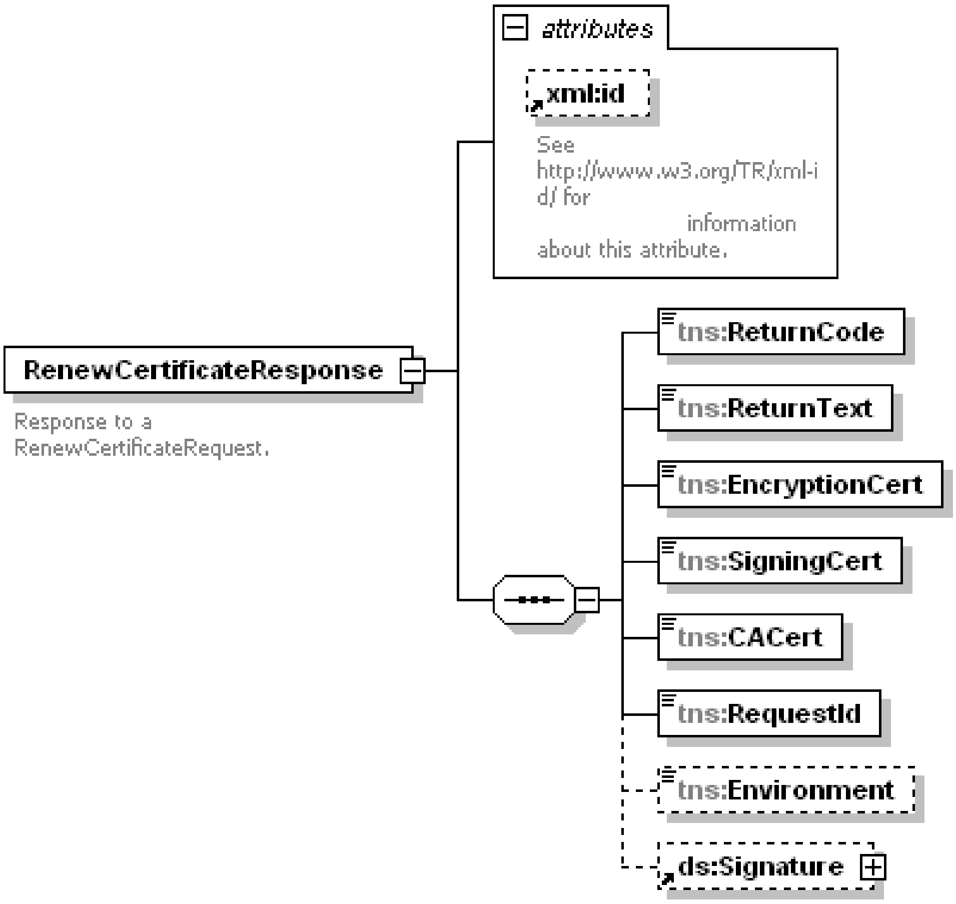
The RenewCertficate operation is used to renew customer certificates if the customer has a valid signing certificate. The parameters in the RenewCertificateRequest are as follows:



Sub element	Meaning
CustomerId	See description in CreateCertificateRequest, section 7.2.1.
KeyGeneratorType	See description in CreateCertificateRequest, section 7.2.1.
EncryptionCertPKCS10	See description in CreateCertificateRequest, section 7.2.1.
SigningCertPKCS10	See description in CreateCertificateRequest, section 7.2.1.
Timestamp	The time at which the request was generated. UTC (Zulu) time must be used.
RequestId	String identifying the request. The combination of CustomerId and RequestId must be unique for 3 months.
Environment	A string identifying the environment. The environments relevant for customers are: • production: The production environment issuing certificates to be used with the banking software. • customertest. A test environment for customers to use when they are creating the integration to the PKI service. If the element is not present, the default is “production”.
Signature	An enveloped signature signing the RenewCertificateRequest element. The signature is based on the customer’s existing signing certificate. If the signature does not validate, no certificates will be issued.

All of these parameters are mandatory except Environment.

The return values of the operation are contained in the RenewCertificateResponse element. This element is structurally identical to the CreateCertificateResponse element, see section 7.2.1. There are differences in returncode values ("User not authorized" / "User Locked" instead of "Wrong CustomerId or PIN").

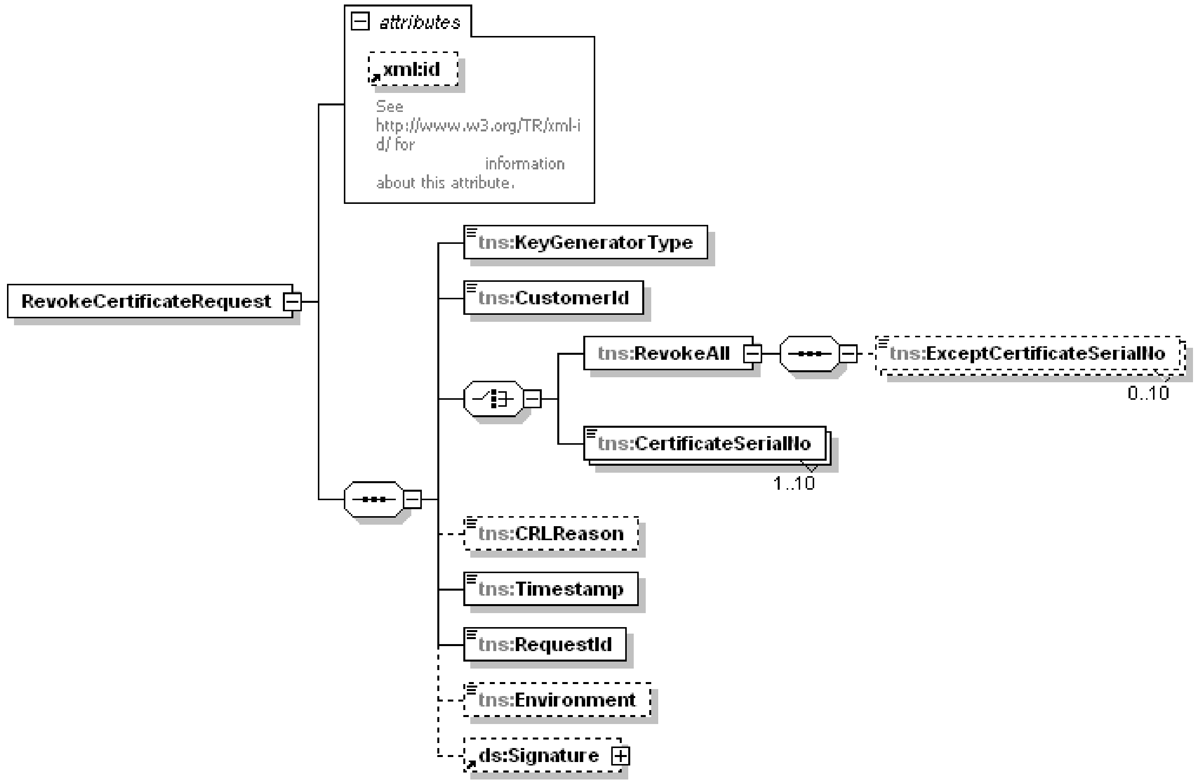


7.2.3. RevokeCertificate

The RevokeCertificate operation is used to revoke certificates. Certificates should be revoked if they are no longer needed by the customer (e.g. if the customer is no longer a customer in the bank) or if the private keys of the certificates are compromised. The certificates will be revoked immediately by the PKI service. Instead of calling the RevokeCertificate operation, it is also possible for the customer to call the bank customer centre and ask for certificate revocation.

Note: Since the certificates are issued in pairs (one encryption certificate and one signing certificate), they are also revoked in pairs. This means that if a request indicates revocation of an encryption certificate, the corresponding signing certificate will also be revoked and vice versa.

The parameters are contained in the RevokeCertificateRequest, which holds the following sub elements:

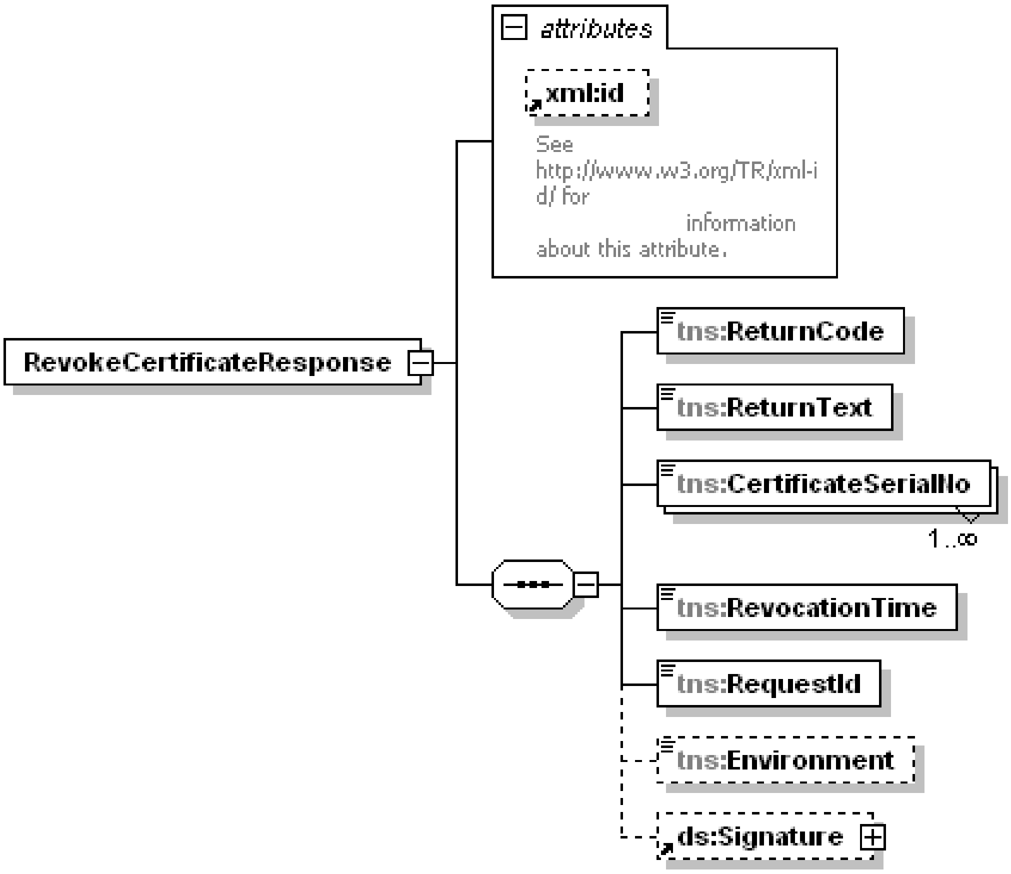


Sub element	Mandatory	Meaning
KeyGeneratorType	M	Information about the mechanism used to generate the keys. Can be 'HSM' or 'software'.
CustomerId	M	See description in CreateCertificateRequest, section 7.2.1.
RevokeAll	O	This element (if present) indicates that all of the customer's certificates (of the type KeyGeneratorType) should be revoked. The optional sub element ExceptCertificateSerialNo can be used to indicate certificates that should not be revoked.
CertificateSerialNo	O	Contains the serial number of the certificate(s) to be revoked. Up to 10 CertificateSerialNo elements may be present.
CRLReason	O	A reason code describing why the certificates are to be revoked. The allowed reason codes are listed in appendix C.
Timestamp	M	The time at which the request was generated. UTC (Zulu) time must be used.
RequestId	M	String identifying the request. The combination of CustomerId and RequestId must be unique for 3 months.

Environment	O	A string identifying the environment. The environments relevant for customers are: • production: The production environment issuing certificates to be used with the banking software. • customertest. A test environment for customers to use when they are creating the integration to the PKI service. If the element is not present, the default is "production"
Signature	M	Enveloped signature signing the RevokeCertificateRequest element. Must be based on the customers signing certificate.

One of the RevokeAll or CertificateSerialNo elements must be present.

The RevokeCertificate operation returns a RevokeCertificateResponse element, which contains the following sub elements:

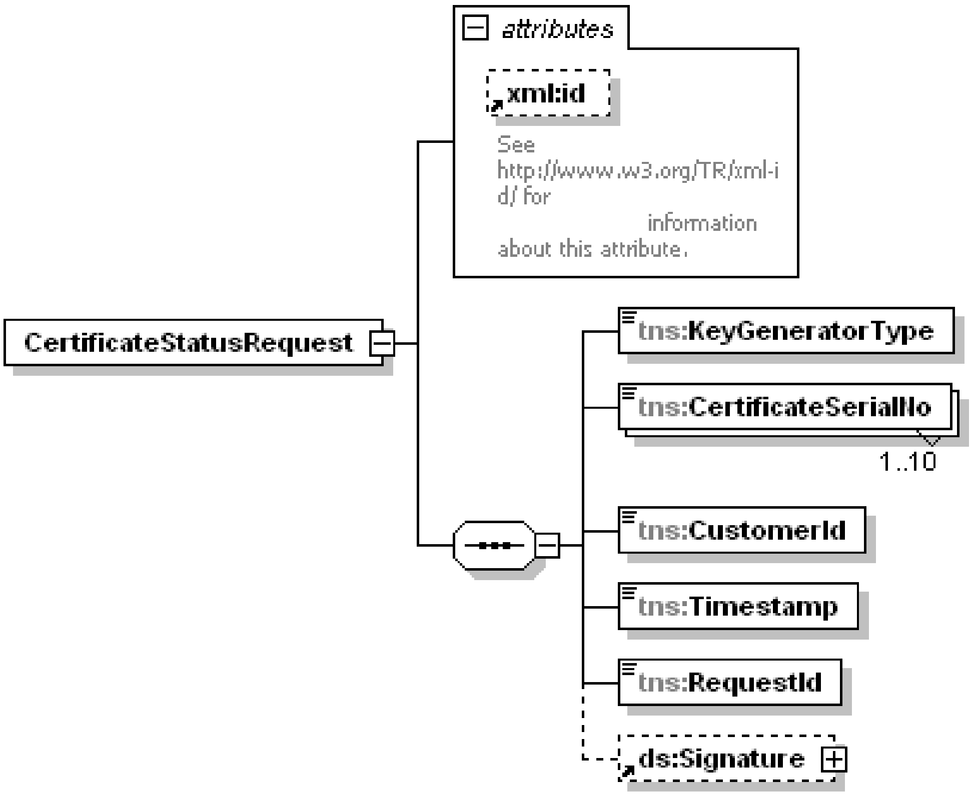


Sub element	Meaning
ReturnCode	The return code of the operation call. If the code is '00', all went well. The possible return codes are listed in appendix A.
ReturnText	See description in CreateCertificateResponse, section 7.2.1.

CertificateSerialNo	Serial number of certificate revoked. There will be one CertificateSerialNo element for every certificate revoked.
RevocationTime	The time from which the certificates is invalid. UTC (Zulu) time will be used.
RequestId	String identifying the request. The RequestId from the request is returned.
Environment	A string identifying the environment. The string sent in the request is echoed back to the customer.
Signature	An enveloped signature signing the RevokeCertificateResponse element. The signature is based on bank signing certificate. The customer should always verify the signature. If the signature does not validate, the result returned should not be trusted, and the bank should be contacted.

7.2.4. CertificateStatus

This operation is used to get the status of a number of certificates. The idea behind the operation is that the customer software should automatically call it regularly (e.g. every day) to check, if the certificates should be renewed. It is also possible for the customer software to look inside the certificate to see the expiry date, but some customers may find it easier to call the service. The CertificateStatusRequest element has the following sub elements:

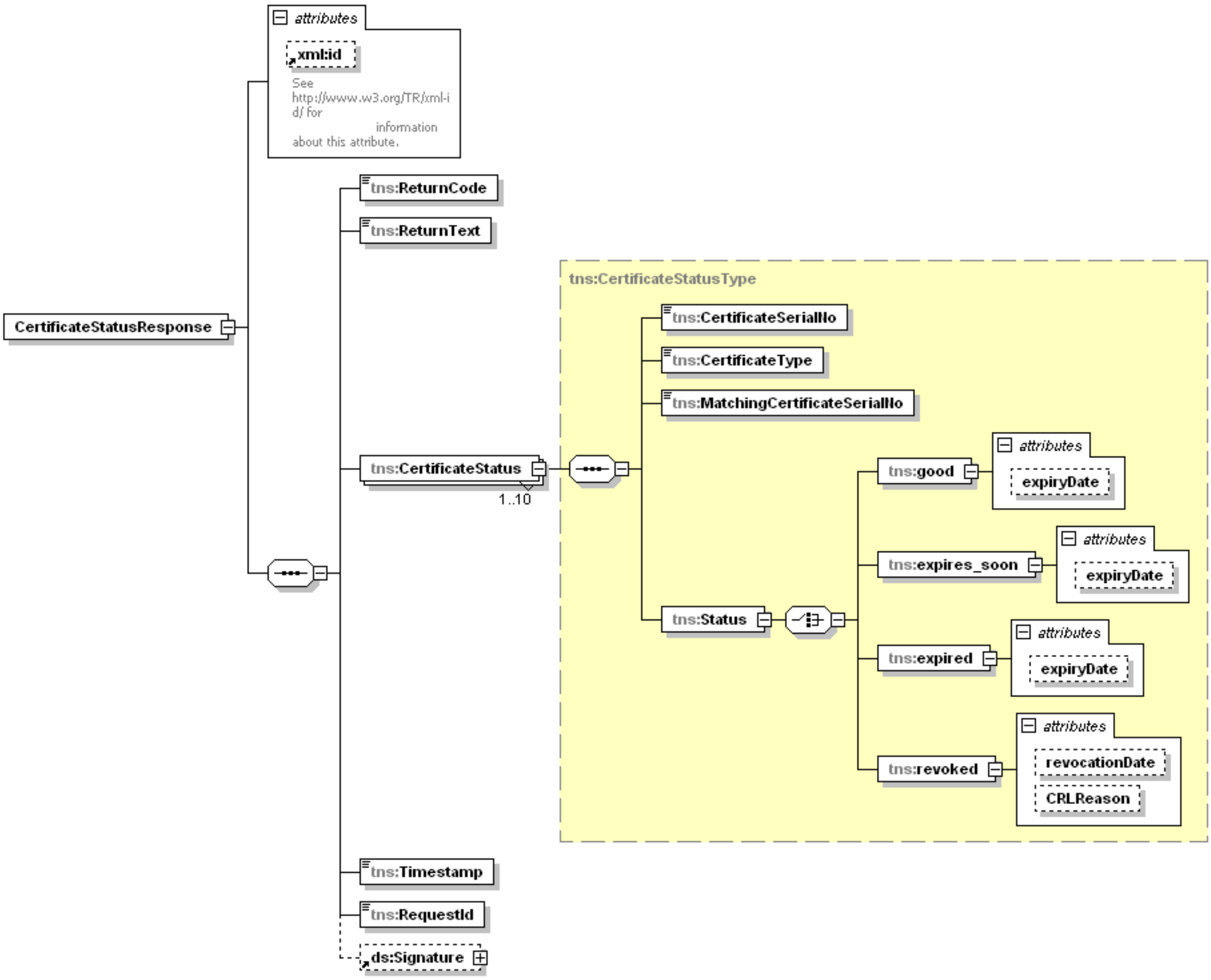


Sub element	Meaning
KeyGeneratorType	Information about the mechanism used to generate the keys. Can be ‘HSM’ or ‘software’.

CertificateSerialNo	The serial number of the certificate. Up to 10 CertificateSerialNo elements can be present.
CustomerId	See description in CreateCertificateRequest, section 7.2.1.
Timestamp	The time at which the request was generated. UTC (Zulu) time must be used.
RequestId	String identifying the request. The RequestId from the request is returned.
Signature	Enveloped signature signing the CertificateStatusRequest element. Must be based on the customers signing certificate.

All of the sub elements are mandatory.

The operation returns a CertificateStatusResponse element, with the following attributes:



Sub element	Meaning
ReturnCode	The return code of the operation call. If the code is ‘00’, all went well. The possible return codes are listed in appendix A.
ReturnText	See description in CreateCertificateResponse, section 7.2.1.
CertificateStatus	Element holding the following sub elements: • CertificateSerialNo: The serial number of the certificate. • CertificateType: either “signing” or “encryption”

	• MatchingCertificateSerialNo: If CertificateType is "signing" then this field contains the serial number of the "encryption" certificate (and vice versa). • Status: element containing one sub element, see the description in the following table.
Timestamp	A timestamp indicating at what time the status was taken. UTC (Zulu) time will be used.
RequestId	String identifying the request. The RequestId from the request is returned.
Signature	An enveloped signature signing the CertificateStatusResponse element. The signature is based on the bank signing certificate. The customer should always verify the signature. If the signature does not validate, the result returned should not be trusted, and the bank should be contacted.

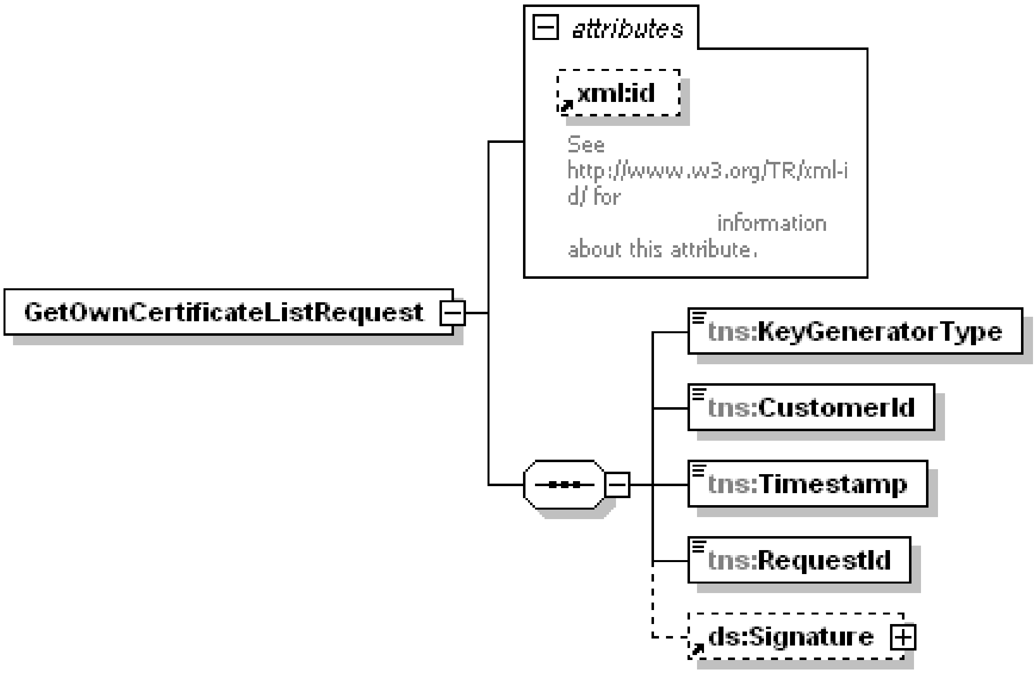
The Status sub-element within the CertificateStatus element contains one of the following sub elements:

Sub element	**Meaning**
good	The certificate status is good. The element has an attribute called expiryDate containing the time of expiry.
expires_soon	The certificate is still valid but must be renewed. The element has an attribute called expiryDate containing the time of expiry.
expired	The certificate has expired. The element has an attribute called expiryDate containing the time of expiry.
revoked	The certificate has been revoked. The element has two attributes • revocationDate containing the time of revocation. UTC (Zulu) time will be used. • CRLReason containing the reason for revocation. The list of possible values can be found in appendix C.

7.2.5. GetOwnCertificateList

This operation returns the list of serial numbers of all certificates with the following statuses : "good", "expires_soon", "revoked" and "expired" owned by the customer, including their statuses. The list includes all certificates issued to the customer of type "KeyGeneratorType". The format of the list is similar to that of the operation CertificateStatus.

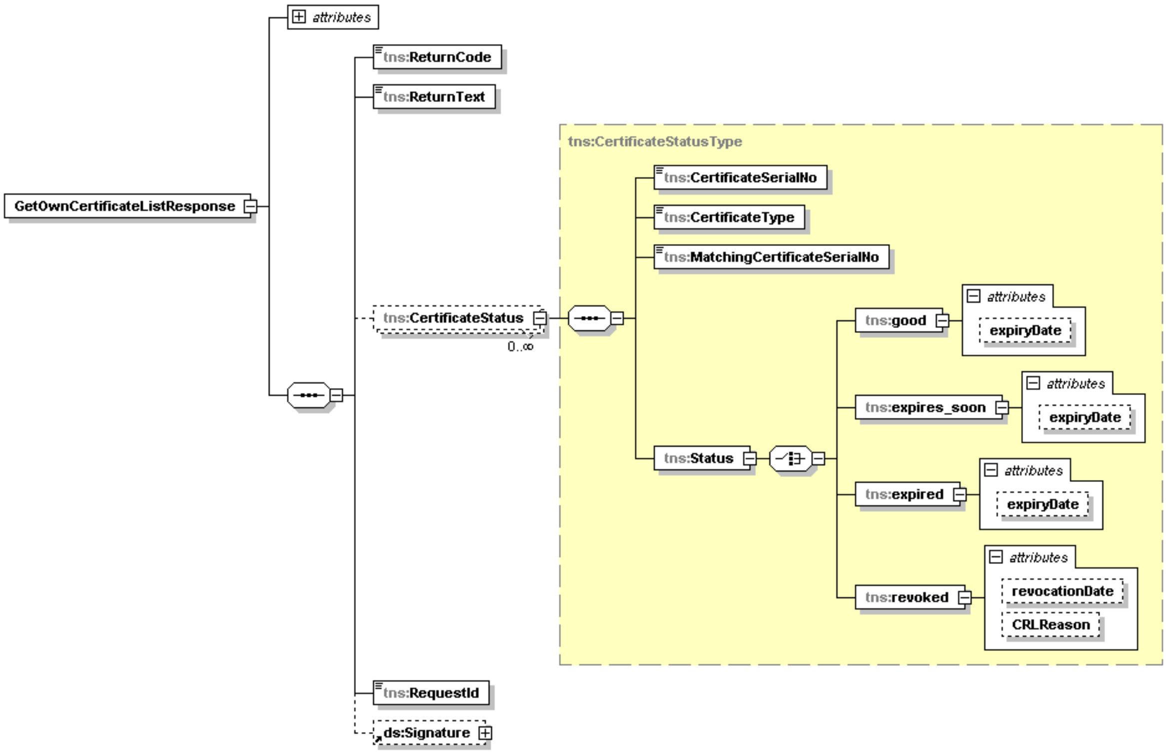
The GetOwnCertificateListRequest element contains the following sub elements:



Sub element	Meaning
KeyGeneratorType	Information about the mechanism used to generate the keys. Can be 'HSM' or 'software'.
CustomerId	See description in CreateCertificateRequest, section 7.2.1.
Timestamp	The time at which the request was generated. UTC (Zulu) time must be used.
RequestId	String identifying the request. The RequestId from the request is returned.
Signature	Enveloped signature signing the GetOwnCertificateListRequest element. Must be based on the customers signing certificate.

All of the sub elements are mandatory.

The GetOwnCertificateListResponse element contains the following sub elements:



Sub element	Meaning
ReturnCode	The return code of the operation call. If the code is '00', all went well. The possible return codes are listed in appendix A.
ReturnText	See description in CreateCertificateResponse, section 7.2.1.
CertificateStatus	See description in CertificateStatusResponse, section 7.2.4.
RequestId	String identifying the request. The RequestId from the request is returned.
Signature	An enveloped signature signing the GetOwnCertificateListResponse element. The signature is based on bank signing certificate. The customer should always verify the signature. If the signature does not validate, the result returned should not be trusted, and the bank should be contacted.

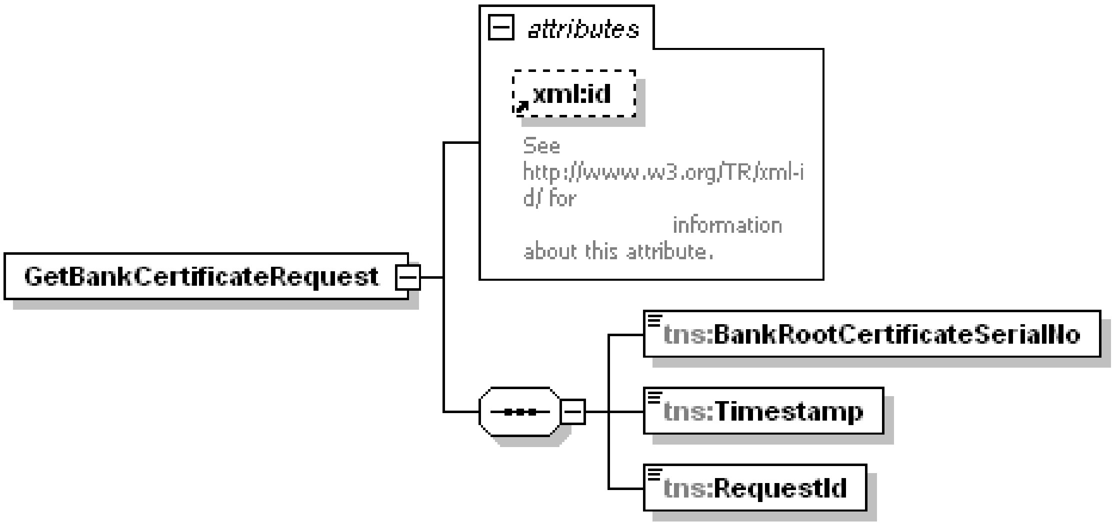
7.2.6. GetBankCertificate

The GetBankCertificate operation returns the current bank certificates. The operation is used to keep the bank certificates kept by the customer up-to-date. The customer software is supposed to automatically call the service at regular intervals and update the bank certificates stored by the customer.

In order to be able to verify the signature on the response, it is necessary for the customer to have a valid bank root certificate. Since the root certificate will also be updated over time, it is also included in the reply.

The operation is **not** supposed to be used in situations where the customer does not have a valid bank root certificate. In such situations the customer must obtain the bank root certificate by other means (maybe on a CD obtained from the bank, the exact mechanism for initial root certificate distribution is yet to be decided).

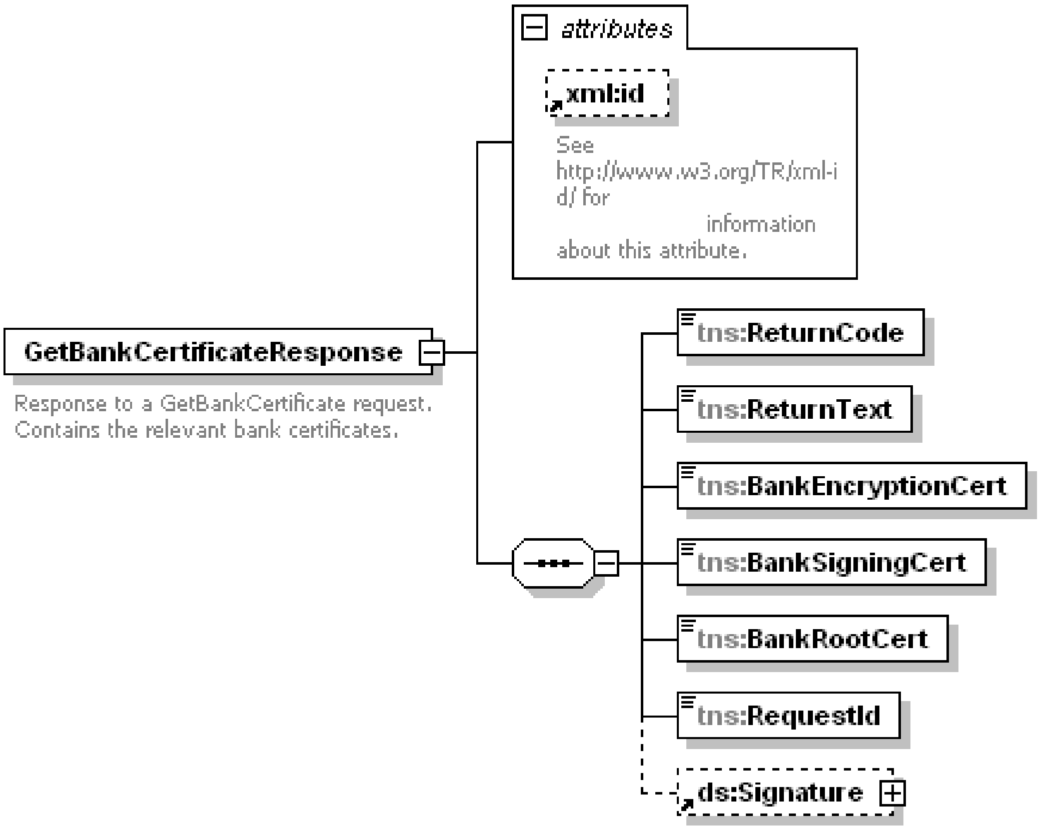
The GetBankCertificateRequest element has the following sub elements:



Sub element	Meaning
BankRootCertificateSerialNo	Serial number of the newest bank root certificate trusted by the customer. The parameter is necessary in order for the bank software to identify the signing certificate to sign the response.
Timestamp	The time at which the request was generated. UTC (Zulu) time must be used.
RequestId	String identifying the request. The RequestId from the request is returned.

All of the sub elements are mandatory.

The GetBankCertificatesResponse element contains the following sub elements:



Sub element	Meaning
ReturnCode	The return code of the operation call. If the code is '00', all went well. The possible return codes are listed in appendix A.
ReturnText	See description in CreateCertificateResponse, section 7.2.1.
BankEncryptionCert	The newest base64 encoded bank encryption certificate issued under BankRootCert.
BankSigningCert	The newest base64 encoded bank signing certificate issued under BankRootCert.
BankRootCert	The current base64 encoded bank root certificate.
RequestId	String identifying the request. The RequestId from the request is returned.
Signature	An enveloped signature signing the GetBankCertificatesResponse element. The customer should always verify the signature. If the signature cannot be verified, the result returned should not be trusted, and the bank should be contacted. In addition to verifying the signature itself, the certificate used should always be checked against the root certificate indicated in the request. The bank certificate used for signing is included in the signature block and is the newest bank signing certificate issued under the root certificate indicated by the serial number in the request.

	It is not necessarily the same as the certificate in the field "BankSigningCert". This will be the case when a new root certificate (and corresponding tree) exists.

Appendix A: Error codes

The following is a list of error codes and descriptions of the error.

General errors (shared between operations):

Error Code	Description
00	No error. Operation went OK.
01	Malformed XML.
02	Invalid signature.
03	Decryption error.
04	Signing certificate expired.
05	Encryption certificate expired.
06	Missing signature.
07	Schema validation error.
08	Signing certificate not valid or trusted.
09	Input error.
10	Other backend error.
11	User not authorized.
12	User locked.
99	Unknown error.

CreateCertificate:

Error Code	Description
20	Wrong CustomerId or PIN (or other PIN error)

RevokeCertificate:

Error Code	Description
30	Some certificates could not be revoked (revoked certs are listed).

GetBankCertificates:

Error Code	Description
40	Unknown root certificate serial number.

Additional error codes may be added at a later time.

Appendix B: Example XML

a. RenewCertificateRequest example

This appendix shows the process of building a request to the PKI service. The examples have been modified in order to improve readability.

The five XML documents in this example are distributed along with this document as well as examples for the other operations in the PKI service.

The content of the PKCS#10 requests in the requests does not match the certificates returned. Note also that the content of the certificates should not be taken as representative of how the real certificates will look.

i. Unsigned request element

The following is an example RenewCertificateRequest. The request is neither signed nor encrypted.


```xml
<tns:RenewCertificateRequest xmlns:xe="http://www.w3.org/2001/04/xmlenc#"
xmlns:xd="http://www.w3.org/2000/09/xmldsig#" xmlns:tns="http://danskebank.dk/PKI/PKIFactoryService/elements"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:pkif="http://danskebank.dk/PKI/PKIFactoryService">
        <tns:CustomerId>360817</tns:CustomerId>
        <tns:KeyGeneratorType>software</tns:KeyGeneratorType>
        <tns:EncryptionCertPKCS10>MIIBnTCCAQYCAQAwXTELMAkGA1UEBhMCU0cxETAPBgNVBAoTCE0yQ3J5cH
RvMRIwEAYDVQQDEwlsb2NhbGhvc3QxJzAlBgkqhkiG9w0BCQEWGGFkbWluQHNlcnZlci5leGFtcGxlLmRvbTCBnzAN
BgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAr1nYY1QrIl1ruB/FqICRrr5nvupdIN+3wF7q915tvEQoc74bnu6b8lbbGRMhzd
zmvQ4SzFfVEAuMMuTHeybPq5th7YDrTNizKKxOBnqE2KYuX9X22A1Kh49soJJFg6kPb9MUgiZBiMlvtb7K3CHfgw5Wa
gWnLl8Lb+ccvKZZl+8CAwEAAaAAMA0GCSqGSIb3DQEBBAUAA4GBAHpoRp5YS55CZpy+wdigQEwjL/wSluvo+Wjtpv
P0YoBMJu4VMKeZi405R7o8oEwiPdIrrliKNknFmHKlaCKTLRcU59ScA6ADEIWUzqmUzP5Cs6jrSRo3NKfg1bd09D1K9rs
QkRc9Urv9mRBlsredGnYECNeRaK5R1yzpOowninXC</tns:EncryptionCertPKCS10>
        <tns:SigningCertPKCS10>MIIBXjCByAIBADAfMQswCQYDVQQGEwJGSTEQMA4GA1UEAxMHZXQgbmF2bjCB
nzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAn4fmoxL7BUtZAcj7I0XjvhJshYJRg4ggYe76Mmtmkm7xjEJv2gGLoE5Y
glaJJauGncFMKo3/OoKJ92XVDAwZg6SYj5cXnsQR3rDo699occzvLuu7UuaNcXw3sd4GtfdoPveKQCbKpp9xXPCydFp9
syT7WAk2L268euedUC4g938CAwEAAaAAMA0GCSqGSIb3DQEBBQUAA4GBABwurQs6f7Ue6Y0NhYPfdWo3qBRXnB
o/d3Wt2yiI2NkFwizbq/ZsJ0wDQ7W4Dg9hRndROjOZTmnYeNKRSJOYHzGrLbgZto7mezdNsW2QL56CsdAh7U7jYo4Of
gdTyc8XQmnsYbspRShV28gOkKgfQCZ3A/oAYkmICEC83ZhkG+1o </tns:SigningCertPKCS10>
        <tns:Timestamp>2022-05-20T12:20:19Z</tns:Timestamp>
        <tns:RequestId>398</tns:RequestId>
</tns:RenewCertificateRequest>
```

ii. Signed request element

This is an example in which the RenewCertificateRequest has been signed:


```xml
<tns:RenewCertificateRequest xmlns:pkif="http://danskebank.dk/PKI/PKIFactoryService"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xe="http://www.w3.org/2001/04/xmlenc#"
xmlns:xd="http://www.w3.org/2000/09/xmldsig#" xmlns:tns="http://danskebank.dk/PKI/PKIFactoryService/elements">
  <tns:CustomerId>360817</tns:CustomerId>
  <tns:KeyGeneratorType>software</tns:KeyGeneratorType>

<tns:EncryptionCertPKCS10>MIIBnTCCAQYCAQAwXTELMAkGA1UEBhMCU0cxETAPBgNVBAoTCE0yQ3J5cHRvMRI
wEAYDVQQDEwlsb2NhbGhvc3QxJzAlBgkqhkiG9w0BCQEWGGFkbWluQHNlcnZlci5leGFtcGxlLmRvbTCBnzANBgkqhk
iG9w0BAQEFAAOBjQAwgYkCgYE
Ar1nYY1QrIl1ruB/FqICRrr5nvupdIN+3wF7q915tvEQoc74bnu6b8IbbGRMhzdzmvQ4SzFfVEAuMMuTHeybPq5th7YDrTNi
zKKxOBnqE2KYuX9X22A1Kh49soJJFg6kPb9MUgiZBiMlvtb7K3CHfgw5WagWnLl8Lb+ccvKZZI+8CAwEAAaAAMA0GC
SqGSIb3DQEBBAUAA4GBAHpoRp5YS55CZpy+wdigQEwjL/wSluvo+WjtpvP0YoBMJu4VMKeZi405R7o8oEwiPdIrrliKNk
nFmHKlaCKTLRcU59ScA6ADElWUzqmUzP5Cs6jrSRo3NKfg1bd09D1K9rsQkRc9Urv9mRBlsredGnYECNeRaK5R1yzp
OowninXC</tns:EncryptionCertPKCS10>

<tns:SigningCertPKCS10>MIIBXjCByAIBADAfMQswCQYDVQQGEwJGSTEQMA4GA1UEAxMHZXQgbmF2bjCBnzANB
gkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAn4fmoxL7BUtZAcj7I0XjvhJshYJRg4ggYe76Mmtmkm7xjEJv2gGLoE5YgIaJJa
uGncFMKo3/OoKJ92XVDAwZg6SYj5cXnsQR3rDo699occzvLuu7UuaNcXw3sd4GtfdoPveKQCbKpp9xXPCydFp9syT7W
Ak2L268euedUC4g938CAwEAAaAAMA0GCSqGSIb3DQEBBQUAA4GBABwurQs6f7Ue6Y0NhYPfdWo3qBRXnBo/d3Wt
2yiI2NkFwizbq/ZsJ0wDQ7W4Dg9hRndROjOZTmnYeNKRSJOYHzGrLbgZto7mezdNsW2QL56CsdAh7U7jYo4OfgdTyc8
XQmnsYbspRShV28gOkKgfQCZ3A/oAYkmICEC83ZhkG+1o </tns:SigningCertPKCS10>
  <tns:Timestamp>2022-05-20T12:20:19Z</tns:Timestamp>
  <tns:RequestId>398</tns:RequestId>

<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>hzgI5MV+vAsNCdMPnDSjXP8X8D0NRypV5dkpWW7+VKE=</DigestValue>
              </Reference>
        </SignedInfo>
           <SignatureValue>
        GI+qRm4tHaEI5viuMtqp8QtHUwwSrnUA+8YJ5fBLSxN7cRelNwDmBxxkzj5wtvNINf6BncGtfM7FMlyUg4fzQ/SB69
        E0ssF17CRdsvQtYBvMZqu7jerQeTarR9nVvMuoIT0wFqjiEvkocTYRUVCpHSo26R0j19LUvSN27fSnOGCh7UZZy
        Y7jpn2rn0uuqW8IDcqiGY8zUU1/QdAiGsWLLtALQIkWIsHdulu9QABWfZsugGs1Yd5Ok+qXswrMyksjja0Mz1Zud4I
        cBu/xFnobF+QsyvrxHYMuOwFc/Fi1d8n8poqHvII7I4pYqPyTMG5fLunAN0kKtBd3Ad/pRfa/Sw==</SignatureValue
        >

        <KeyInfo>
              <X509Data>
                    <X509Certificate>MIIDdzCCAl+gAwIBAgIHBmQuU/N3CTANBgkqhkiG9w0BAQsFADCBkjEQMA4GA1
                       UEAxMHREJHQ0FEQjELMAkGA1UEBhMCREsxEzARBgNVBAcTCkNvcGVuaGFnZW4xEDAOBgN
                       VBAgTB0RIbm1hcmsxGjAYBgNVBAoTEURhbnNrZSBCYW5rIEdyb3VwMRQwEgYDVQQLEwtEYW
                       5za2UgQmFuazEYMBYGA1UEBRMPNjExMjYyMjgyMjMwMTAyMB4XDTEwMTAwODA5NDQ1NFo
                       XDTEyMTAwODA5NDQ1NFowgZQxJjAkBgNVBAMTHUhBTIMgUC5IQU5TRU4gT0cgSlIUVEUgSEF
                       OU0VOMQswCQYDVQQGEwJESzEnMCUGA1UEChMeRVNCSkVSyBYWFhYWFhYWFhYWFggI
                       CAgICAgICAuMTQwMgYDVQQFEytTRS1OUi9EQUJBOjAwMTQ5ODE5MTgtQUdSOjY2MDY4NC1
                       VU1I6MzYwODE3MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDMZBElJ7Jh9Ogd/nm9zL
                       uahfYaz2rJ8u5kHj20hIeY1EaG2eTsPBa1p2UhbgaM1/wdIujCu/zz8wg3DW1G7JawB3bS5nIQ+7A0i/
                       OjA2A7QSF6kPZNu6LZIBy7gDKujpihc8vPutPMXyvqT7zoeILBqroPZodCgB++y01mocC8ywIDAQAB
                       o1IwUDAfBgNVHSMEGDAWgBRc1ZAUHzhAuhRy3UBGMj0IIqOdozAdBgNVHQ4EFgQUGNq9ncto
                       QOZhO9gUvNneZdMaw2IwDgYDVR0PAQH/BAQDAgbAMA0GCSqGSIb3DQEBCwUAA4IBAQBvy2
                       ZKbkLzLChIMnb4o+3sGSiikQycoClc1owJ14HisDKKgpYeGWMmamaGZEkVFI9Y+gHl6ZNvtO2oyyv
                       JrlSPBhK/RJAOLSrSoIqEqa15UoDIxgPo5zu+dguN57g47JsrQG0hFVBlItKKOy2whcGe8Qh1bjnunib
```

```
                mL0XdSgmJ4a6GzPR+YlWu+Lag7tCii+1pYMSh1bWF5h3N9MLZ+85IAv5TJl8407FLnbBypRkOdkH
                dKAyE70cJGI975lmDWaHQ0U98AA2/RQfQzDdvsvEONNRpJu98x9AMJzBGsS0iFTd4y/X9qI/4r9ftj/9
                Swu73ayFvkO2T6/tWPAHJNsJ9</X509Certificate>
            <X509IssuerSerial>
                <X509IssuerName>serialNumber=611262282230102,  OU=Danske Bank, O=Danske Bank
Group, ST=Denmark, L=Copenhagen, C=DK, CN=DBGCADB</X509IssuerName>
                <X509SerialNumber>1799000000001801</X509SerialNumber>
            </X509IssuerSerial>
        </X509Data>
    </KeyInfo>
</Signature>
</tns:RenewCertificateRequest>
```


The example uses the standard entire document enveloped signature reference (<Reference URI="">), and the signature covers the RenewCertificate request element.

It is also allowed to use an id attribute (eg: id="G972dd040-2D") on the RenewCertificateRequest element and a reference in the signature pointing to the id (eg: <Reference URI="#G972dd040-2D">).

iii. Encrypted request element

In the next example, the signed RenewCertificateRequest has been encrypted:


```
<xenc:EncryptedData 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"/>

    <dsig:KeyInfo xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
        <xenc:EncryptedKey Recipient="name:DanskeBankCryptCERT">
            <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
            <dsig:KeyInfo>
                <dsig:X509Data>
                    <dsig:X509Certificate>MIIDmDCCAoCgAwIBAgIFAQjonrMwDQYJKoZIhvcNAQELBQA
                    wgcQxEDAOBgNVBAMTB0RCR1JPT1QxCzAJBgNVBAYTAkRLMRMwEQYDVQQHEw
                    pDb3BlbmhhZ2VuMRAwDgYDVQQIEwdEZW5tYXJrMRowGAYDVQQKExFEYW5za2U
                    gQmFuayBHcm91cDEaMBgGA1UECxMRRGFuc2tlIEJhbmsgR3JvdXAxGDAWBgNVBA
                    UTDzYxMTI2MjI4MTEzMDAwMTEJMAcGA1UEBBMAMQkwBwYDVQQqEwAxCTAHBg
                    NVBAwTADEJMAcGA1UEERMAMB4XDTEwMDgwNTEyMDAwMFoXDTExMTExOTEy
                    MDAwMFowgZkxETAPBgNVBAMTCERCR0NSWVBUMQswCQYDVQQGEwJESzETM
                    BEGA1UEBxMKQ29wZW5oYWdlbjEQMA4GA1UECBMHRGVubWFyazEaMBgGA1UE
                    ChMRRGFuc2tlIEJhbmsgR3JvdXAxGjAYBgNVBAsTEURhbnNrZSBCYW5rIEdyb3VwM
                    RgwFgYDVQQFEw82MTEyNjIyODQ0MzAwMDMwgZ0wDQYJKoZIhvcNAQEBBQADgY
                    sAMIGHAoGBAKlp7a01sKY9GeLvi3ilenNTCxRNEF5rM+lyFkeOf4cGI9AAfrqQLM5zEsc
                    6svvUIVgiPoHhaupdnHhJSV1Ckq7ru9KHImwvLJc9meIKj4O1fbM5gYSKDOrqEjKVHIcq
                    deRl8lLGj9OAZfeuQvEXpxto/qbE3stwK/11SP55GeeBAgEDo0AwPjARBgNVHQ4ECgQI
                    REJHQ1JZUFQwGQYDVR0jBBIwEIAOwsbEw/b2wvb39ff1xcQwDgYDVR0PAQH/BAQD
                    AgQwMA0GCSqGSIb3DQEBCwUAA4IBAQCaYHHemIJ5cBfSZ1XdbtjTLC41591Ngf5Nu
                    0+nRaF8zKCoJ2M44m9EC7ecNMLwyCm+4pZYlatePhBJQH3ScGVhbg2EVb1B3WDfB
                    Ki8BMEDS5nGS9PLQ0b2gERPHWox3XFhJoN2wWyowZqYfLZPR3h/3au6TICGfInCL5
                    9lefd998Ae1K7Qbj9/KcAjDWUXcdmzxgG5sbHgW8jHPiUmm8ecPxqhF+4Q6ocgN2Ny1
                    9hsb9ugaPAJRlqpVGscfnwMECgvyU/8dWJ1Qe3a+Rmmj6buS/K+6AA4xiguGBOE9w/b
                    AX6csygMnPg44ylaNPiMPUnsf/Zzsj9tnpEyFnNN/w12</dsig:X509Certificate>
                </dsig:X509Data>
            </dsig:KeyInfo>
            <xenc:CipherData>
                <xenc:CipherValue>
                FkIKQYPuPwOcTYjO97hVVZFvtgvTZjXnPraJkF8HWpwhhP5YAuqOLM/MpPFBLSVFPz44HW
                Ai3GpOg6aUXawAEq15vSW6nW+zD4/J0X9Il0hPYis3DA9zfZ7aRBf2h/3/dwJ93dbY6E4HTmW
                3yjkeBXAvAZZtPWNcyEhYNYTr1cFvPeb0ddnmeWv3tGIGRs6vgW55dFiFMcKCQzO1cbnSX1
                YYDyspVuC8HZAT9D1s6tG7ygvRKJp8WbFr+Orm+OC8hzy8/xDHhj8JLPZScpDd0AfTmDXSD
```

U+oK3MYSNlWfAFZfUEYK3lfUlrj2R9K305RD6GQfZX0bZy8kXueWMsMdA==</xenc:CipherValue>
</xenc:CipherData>
</xenc:EncryptedKey>
</dsig:KeyInfo>

<xenc:CipherData>
<xenc:CipherValue>AxAGIbMMCIrSiNY/6XfOp80t+vPjcCpJAKfoE3R7PkAbdPny4MOeSCU1b+Hf00J6ub
e0dhFO6LijbHEVvmzsIm9fJOBFtc5p3hb0znVCqTDNN3c7yKR6JxnQft+y/S2k5rdHi0+9fFzY9yzAg1XCXH
ZQaC3grm8CsPqoMhWYAqAFkdIAak3H9jMDXbgLFZvdiCSj9Kp7LzwjXZ3OZcl/sTzVbGqseO0W7v2h9liW
kGVuHZe/EO8AqKwhZ2R5gEJlgCZi/BKFFJH7dAByUaL4D1egvj3jntjBDl2aZAHDmGvvGSbjyM5t2cqarLY
HmX/wJpf52QoWWHtfaSphIU62mMV945bPR/07tWwWejSyM4K2p3vDqv4Fihm0tsdzpcz05WLzrhkKmu2
SNnKyr1nP3Dsuuqb35Xk31AG+FNrT6J1lJszww6iMDPQhjQBJlgpKT3RNKJNMsqu8Vai2TUYWI9B/anMR
6cmlrl/AQ4A/4TZJl9DXF0J6t86UIOim8Cm52jmphj75i+lyw3UZ7gO+9Bx+CnokUxyYq2wnoBqLA51iLB3zC
5OTKbynQCLFWDa9LArOzaFrkiewVLL/82Bjj9e+cfwE3TMK2Yqgz+iT7d2YMTV+HpiDgL0yqSyefA0eVlV
WKACu570zVl66qT4pk0Qs5xq0bt0dA+mDRQpijfGam3JmjqEtl8WUDTPU50jX+P7e/TwQACfTxVJiUyH1y
3qT6c5Qin6gqE60EvZ/OSH97UQ6luu/TSGNOvWuQhVWHVauGyJwHdfPsKppLfAZMeNyc9dZpBmeGu5/
VtOwrzwBDTDybxjxoSv+sHSU/FAXltAWjo5jN4QinKyAPh83PSHQP28nh8BexnRk+8M1yA5zXB7na6QrAs
2sN15yvOJrzQhgVJDMjkYm3M88PMtcsu7fkkk8TK7ejTXmuQI1HhefDEQWWIK+715QmjXirlmn82dE/Erp
Cbpgo7TtlrNh4WPu01nSEKyRojgh2TzG23CZThfbUJs72FZJAZ1alt5DGGbj4jeeqBfg5KNMPczWxjv1/XjE
2f8vm5Dr7mLx2833/2NB7nWDM4zALlGuApa9GcgIR4vr+rr+h4qc1DSFqsVoyCTXZ6b4YL6kV1n7GH+YH
NPZ8MqWMo3HbdojuPQG4J1uceNtY/9MRhYLkfSmlvJKU7OyOi1BtZzEpr1SOcEmlZMhzlKoIGsIFUEsdN
6PPALJm2dYe+phWSYDHH75eXkn3BEg5HhH+HC1VUuBaHEbokuQ+BvRxNgGv70XoDSFIvc2toGvPjY/
gWkKtVeYVcqBiwodcq18PUup80wbF6gx7jaqLonjzHnABjI00bfZeaSsycQj+4rJvXBcx2blthfMumrvk27frxiF
BLdEkH0tbGtbfBM1um+X2pHSLotbtevdwVv05OQoB51p3Hm6m9a93Joef56lhxUHyxbAc5Kf1Jeh53UD4l8
OpJ8QXHC4D5+nAMoSk5X1Dpdx6aEGrnE23yGPPn7XE5ykIl4qkfqCJnV6VqWLTNnneC/m99o23iL1p31t
Vck7eP65a3OC9q0zgvTmv0RZAU4EJbJ4PYJ4zyJNw0ZqKDpFUyGLDtJuQiKMsrfUwg/n3tHDtiPFV20tFh
Rmkpal6y+Y5ZGDIqSBJtQ49/P4vFn+GhmytPNezhBLIZlprelUdxkZeNVgDgA2yZZa6CWP/YpP8xlnZReW
A65fai8KYsNNlArpbP3yHKj2XFzU3S23eVONMrS1gltVQ+nGP2B05nEXOG6u9YjDVeSVH6ZkDq+4QVLJ
5st3IfbrmjCV9u7bjLJ+Vo4WRGz2LUvQdt6/Wf0onLRSAKPrBmH9N8AdPpYMcRd2/N7SA/lPM+yNk4+jyWf
ySpDZnxo4r2r65lGu1i540Ra0F+H3wPrYcv7VXQo0yGWKVyJnTq96E03ieEPgsVX9Kt0hKrdh+53CM/FDjp
K/HslyF5s8+AqZTRfcfYO1qr0ssEJdDtK3pWyAMbtRVJMBH9R3w93vSMHzEhudwTHisgNThCrz9qymhZz
1cao3JMQeOXnJHR5E9viiUVmxS/fCQn1T9RN6zwltPCHSTs9wp5LSzhoDKGYJ2y39NG780s7uecMSyF
U6ucKOhi1mP+np4PBDzslrxy2Qm59aGMPsLmv2qGQ4tq3BpGR5ns9SbA24/7jVdv0YwA0A6RR5+Yp5N
5H6F0QCr0mUiqduHPq7XavmTX8YVGpWB/WJlpexqD0oQ3CrzwnCscy0H3huv6geMrY8Br0/xWn9EAtlJ
wP9A6Td8A7xSBR5/07aghmjgwFIMDOsEwMX/FkYp5f7zvDvV7nhIFpSATs2AlnuZMei+b2H2JDHQMS+W
1/QdJ2DcDrXiuPlFE4oGJWzsZzk4qoX4GGRhH1gZyLJCcYODCNOUwrA1l+rw4U45G7MSHSteDvcJ8qd/
1HWSvMYjFm4NV/Ca91FI395IhGUcz5aQuvrb1wfaM81cjYpqXi33XnVIGOXgtr5tXbSMIgbIgZA9VEbo0yFa
++XYBO+XZnL3UYXcknxQRxYPFKaWJJCCS7aRbpVFMEzSyUPRev8VSrdZxeiUwQ9zWEps3uK3lFLBg
YT5YNFW/bGUPsezsuHW3zpz4E61ZNzBsxQulxF755+BjPMgYDQ/Mqs4KozYYcJzGjaeBIA2HzpwLo+Oz
ZPRw7Yonfylegs5SaDay+lX5OJps0kp0tHUB8tsRvdkGo8yZcKPY4d9jNZl4Hc1kcz+K/WuXx3TnIl7Ob9EK
hRTY9a/+Jh3RXFgwrHBhdGdOThbWiMjy38KzwlyF8ZmFOdkm8meMuscsFQN3bewTcFbUqACqGCXRAj
S0e7fe8CXqgUBt8faTvuXFtHf1N3TFsyJEp89xGfPR8Ew5sTjV2n6w1H2zEN4dj9ZkkdqIwh8kaL2JD4A5Io
mLXEngkv/w3kgPE3Mk1RxWzLTrS/D/rLvLmGHc2XzsGawnQdC2kBuLgbVTNWb8F0GrbIXX84INAO7gz
w/uifMQSpU3jEv++xsZDBUMEptKGEIJcOoMgoTjqtuvOxFp3+Kn+I8IRCA0Iz+/SzrL5THCYxHGBFmzF9O
aIDWOwfShym0Dkhsk9y8Ky9YfEfY1afTayKy1BL/4iQ0gNvJaaTcUXe4LT97vkLWtSCA0XbZdh0THQxfm4
qVzrWRjnr0jPBWL4zeA//JkXUvfOyqZeEPl6Mm+rdFQvQa58Umdyc4au5gIp2NTAGHWz6h1QOjdN+r4llyw
8jCG6fnvCUTz5LPVvJu6njAedt1JIreV+2qhqLh8y4WZ+KNoMwuUujEuGufXYr+Dr4agAyV4IgyOo4YtbXtlX
1A3hp24d0XN2hZEFwMJ7PEJkTk0tgHNwWH77Hnp1woTXLDWJUYFdn3umf2ct1Ph5JuEL4R0ZI8kQ1p2
baTZLhPH804NF4cyz5R0BnewrNd6OzKEcRCQdYj8xpvGQOc5WsScMPbjndRrE9R/MTQR5SMt3ODiOy
5yaX+v+G4haEzSb/PlTI5IIij2oDjcw2IXwPS+g8ny5WPmpc+AQjPjzTGU84QcfqRFANaVEfGEda++cyvKFT
Fcu9kXKP1vMdiBCgn/jq71vrTOMleAwVwdOgY1uly5p7JxVoPp0+q1XmOd8HRbTo7waxBeaT0d0w2rbazj
nWXRz5b+CZhejitmi9fIG6r0t9053Jk8QRLbHSAuokwF99xWGJjuYVCewcOIf6fD1WcLwVqdrpygNJOVXW
55vHQxOZCsw/lr24iSdS7KFqNK3KztxI3tKHn/udn25hddtP1bbbgIr0WCMaZj5Q8aRCKjerykI31fmNc1mBG
7pKAPFL55NGdabA1gFibq1euRf+O0Uma+bYAsMYvkc2H4Pj86dog1SyZI8oUejClC/iESyh5yTB+ucH+z+8
k6dMRjUccYeYP3rwKCBUOqp3zRMu55MSAiTqV3uEIWNpoWY6RCmiUqL319uwaw8NGy4A1aEf4M1Ku
N8/Q3aZYNFiQP7KOB1A2UZrH6fhcZBFAkyUh/FLroO5Rcxp5JWNlr/jZi0NU/cG3yE6W7zlbugvGI+XNV91/
yh/KKTTBCZXmpDdUjXfSjNU5OGV79TDpEIKBeY40MOQX1a4LPATa7hMRKjwnLEkKSwrqj6kUJctkh0cU
YjbDv3a/S1bN8CTZxqsHFes4MrY6zzZ2WutUCaCuF3GI1FoAH7uqhJoI1P8CdK1op9oPhIn66BG1s9/I8E6
AVp+ei2BX1wNtCpWaP/xwDXZYGJPBEv6pzWfrlNtj8p794C67LsmjCKqUwcAn013FHWICILJozkU2vRFw
hgb1s3mvmcTmoMsrfhWp1iLC03GK2OQD9AWPb2500MjFPhwfCkZH2EtaYoFIZfAexhAkqQOfWQQFHKl
5vK+JxI7bR2QmZovD4b/ebTsk69ytzsiRmyGamCPkCQKSyp+RE3bk9ypsZO4zjw226iAzhsrwuLIHLSGjhF
xNZ4h97gBK0lfmQPil8Nvd6MF5TeUal8ELv7HzPP45cOx1IRfAEygMOjENPJe0xbUbKU4ENY6fwmgc2l4d
JqVokfjFqp09ttyt6FQh/WnOd+d7aeQmwfSlzeFhjJLGROJyWkjM4Uirsja78tW1PHbn5QkVNnYCbu8YowlE/

```
                N7nd/1fXwLMWcCC8ledCdgrjF73rjHiK8amSYV3A/i2MgEh6K/ccWOhLEMf6YzN4wVpK4lzVAdc0+UxcB8
                DSizkKkMNhme9KCNIRLUF4cZnHTHI5LsofY8oRLkmxg/71rL8eoJYjQ9zCuoAWNvt7M2B7uM1TYRwflWh
                w1ElhAHlblFmMOZukBQ+AN67LdJ9tnyICYsV7eoEf4ajww+ttEMuYqNrpzjawgtKY6wR7Ob5mFlk00NgLC
                Bss/NeZ/nYQTOoxF77jR46reHAMWolwxEcb2Eb7/RyWdHu6IVxZbIEQ9HTzUa4ftZ5lxSKdMbxH9gFAcry0
                NzWQeFygbarjvkK5kumcsCJK51xeScJdnafyUWDuGNKks0LVSuMknfHWDRO+JXCW54A5sHy2oiwiITW
                nyajO8gMrt9UySYyR5PBDd/pz1edmlf++TdAls7BboBUtQN1HNhtyIN6WfWzWCluK2cxMqeoh0prFYGLEkl
                aDs05EbnAGvDHw0F4A7uPIF1VXOPLXSmw7HKSPlai2TKu5BTQ1byNGuVaeshPsuGpsw70Y7nLF4DM
                svBUqi2DgVf5cnBoWBWLoJQWk5TT5egklbiCpDGZ2qsBvlBvW0dmlMOMf9HTf/gdYIH9JHmIndKUe+xkw
                9qOtF7xgDOX0WzPN0Rd7k2+MWLTxLNaakUT0KkgeoFqWTI20Uy3y1xce5pDFay9cd0EFG4Ew/dwV2a
                GPhVgcrbjEEykH4H9TQapTJ31CUCJxGKG0BP+FoPEr/Y</xenc:CipherValue>
        </xenc:CipherData>

</xenc:EncryptedData>
```

iv. SOAP request

Next, the encrypted RenewCertificateRequest is inserted in a SOAP message, where the RequestHeader is also added:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:pkif="http://danskebank.dk/PKI/PKIFactoryService">
        <soapenv:Header/>
         <soapenv:Body>
              <pkif:RenewCertificateIn>
                     <pkif:RequestHeader>
                               <pkif:SenderId>360817</pkif:SenderId>
                               <pkif:CustomerId>360817</pkif:CustomerId>
                               <pkif:RequestId>398</pkif:RequestId>
                               <pkif:Timestamp>2022-05-20T12:20:19Z</pkif:Timestamp>
                               <pkif:InterfaceVersion>1</pkif:InterfaceVersion>
                               <pkif:Environment>customertest</pkif:Environment>
                     </pkif:RequestHeader>

                     <xenc:EncryptedData 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"/><dsig:KeyInfo xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"><xenc:EncryptedKey
                               Recipient="name:DanskeBankCryptCERT"><xenc:EncryptionMethod
                               Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-
                               1_5"/><dsig:KeyInfo><dsig:X509Data><dsig:X509Certificate>MIIDmDCCAoCgAwIBAgIFAQjon
                               rMwDQYJKoZIhvcNAQELBQAwgcQxEDAOBgNVBAMTB0RCR1JPT1QxCzAJBgNVBAYTAkR
                               LMRMwEQYDVQQHEwpDb3BlbmhhZ2VuMRAwDgYDVQQIEwdEZW5tYXJrMRowGAYDVQQ
                               KExFEYW5za2UgQmFuayBHcm91cDEaMBgGA1UECxMRRGFuc2tlIEJhbmsgR3JvdXAxGDA
                               WBgNVBAUTDzYxMTI2MjI4MTEzMDAwMTEJMAcGA1UEBBMAMQkwBwYDVQQqEwAxCTA
                               HBgNVBAwTADEJMAcGA1UEERMAMB4XDTEwMDgwNTEyMDAwMFoXDTExMTExOTEyMD
                               AwMFowgZkxETAPBgNVBAMTCERCR0NSWVBUMQswCQYDVQQGEwJESzETMBEGA1UE
                               BxMKQ29wZW5oYWdlbjEQMA4GA1UECBMHRGVubWFyazEaMBgGA1UEChMRRGFuc2tlIEJ
                               hbmsgR3JvdXAxGjAYBgNVBAsTEURhbnNrZSBCYW5rIEdyb3VwMRgwFgYDVQQFEw82MTE
                               yNjIyODQ0MzAwMDMwgZ0wDQYJKoZIhvcNAQEBBQADgYsAMIGHAoGBAKlp7a01sKY9GeL
                               vi3ilenNTCxRNEF5rM+lyFkeOf4cGl9AAfrqQLM5zEsc6svvUIVgiPoHhaupdnHhJSV1Ckq7ru9KH
                               lmwvLJc9meIKj4O1fbM5gYSKDOrqEjKVHlcqdeRl8ILGj9OAZfeuQvEXpxto/qbE3stwK/11SP55
                               GeeBAgEDo0AwPjARBgNVHQ4ECgQIREJHQ1JZUFQwGQYDVR0jBBIwEIAOwsbEw/b2wvb3
                               9ff1xcQwDgYDVR0PAQH/BAQDAgQwMA0GCSqGSIb3DQEBCwUAA4IBAQCaYHHemIJ5cBf
                               SZ1XdbtjTLC41591Ngf5Nu0+nRaF8zKCoJ2M44m9EC7ecNMLwyCm+4pZYlatePhBJQH3ScG
                               Vhbg2EVb1B3WDfBKi8BMEDS5nGS9PLQ0b2gERPHWox3XFhJoN2wWyowZqYfLZPR3h/3au
                               6TICGflnCL59lefd998Ae1K7Qbj9/KcAjDWUXcdmzxgG5sbHgW8jHPiUmm8ecPxqhF+4Q6ocgN
                               2Ny19hsb9ugaPAJRIqpVGscfnwMECgvyU/8dWJ1Qe3a+Rmmj6buS/K+6AA4xiguGBOE9w/bA
                               X6csygMnPg44yIaNPiMPUnsf/Zzsj9tnpEyFnNN/w12</dsig:X509Certificate></dsig:X509Data><
                               /dsig:KeyInfo><xenc:CipherData><xenc:CipherValue>aNMQTnDcsF/TJPWRcI7CHVIYVyTEdH
                               BXBOI3kBI3hYG2alz2ex6uqNDF22MTA9YMC/R3jhEBSSAjytLoFB8CoD3U6hVFnlv/n2O2QjrKt
                               2t8DVu33w5K71lmMFYV084IFmqEbpzpIFS7K3Lm0OMNdKngH7mzx5bOROIl+b/Jbik=</xenc:
                               CipherValue></xenc:CipherData></xenc:EncryptedKey></dsig:KeyInfo><xenc:CipherData><xe
                               nc:CipherValue>AxAGIbMMCIrSiNY/6XfOp80t+vPjcCpJAKfoE3R7PkAbdPny4MOeSCU1b+Hf0
```

0J6ube0dhFO6LijbHEVvmzsIm9fJOBFtc5p3hb0znVCqTDNN3c7yKR6JxnQft+y/S2k5rdHi0+9fF
zY9yzAg1XCXHZQaC3grm8CsPqoMhWYAqAFkdIAak3H9jMDXbgLFZvdiCSj9Kp7LzwjXZ3OZc
l/sTzVbGqseO0W7v2h9liWkGVuHZe/EO8AqKwhZ2R5gEJIgCZi/BKFFJH7dAByUaL4D1egvj3jn
tjBDl2aZAHDmGvvGSbjyM5t2cqarLYHmX/wJpf52QoWWHtfaSphIU62mMV945bPR/07tWwWej
SyM4K2p3vDqv4Fihm0tsdzpcz05WLzrhkKmu2SNnKyr1nP3Dsuuqb35Xk31AG+FNrT6J1IJszw
w6iMDPQhjQBJIgpKT3RNKJNMsqu8Vai2TUYWI9B/anMR6cmlrl/AQ4A/4TZJI9DXF0J6t86UIOi
m8Cm52jmphj75i+lyw3UZ7gO+9Bx+CnokUxyYq2wnoBqLA51iLB3zC5OTKbynQCLFWDa9LAr
OzaFrkiewVLL/82Bjj9e+cfwE3TMK2Yqgz+iT7d2YMTV+HpiDgL0yqSyefA0eVlVWKACu570zVI6
6qT4pk0Qs5xq0bt0dA+mDRQpijfGam3JmjqEtl8WUDTPU50jX+P7e/TwQACfTxVJiUyH1y3qT6c
5Qin6gqE60EvZ/OSH97UQ6luu/TSGNOvWuQhVWHVauGyJwHdfPsKppLfAZMeNyc9dZpBme
Gu5/VtOwrzwBDTDybxjxoSv+sHSU/FAXItAWjo5jN4QinKyAPh83PSHQP28nh8BexnRk+8M1yA
5zXB7na6QrAs2sN15yvOJrzQhgVJDMjkYm3M88PMtcsu7fkkk8TK7ejTXmuQI1HhefDEQWWIK
+715QmjXirImn82dE/ErpCbpgo7TtIrNh4WPu01nSEKyRojgh2TzG23CZThfbUJs72FZJAZ1aIt5
DGGbj4jeeqBfg5KNMPczWxjv1/XjE2f8vm5Dr7mLx2833/2NB7nWDM4zALIGuApa9GcgIR4vr+rr
+h4qc1DSFqsVoyCTXZ6b4YL6kV1n7GH+YHNPZ8MqWMo3HbdojuPQG4J1uceNtY/9MRhYLk
fSmlvJKU7OyOi1BtZzEpr1SOcEmIZMhzlKoIGsIFUEsdN6PPALJm2dYe+phWSYDHH75eXkn3
BEg5HhH+HC1VUuBaHEbokuQ+BvRxNgGv70XoDSFIvc2toGvPjY/gWkKtVeYVcqBiwodcq18P
Uup80wbF6gx7jaqLonjzHnABjI00bfZeaSsycQj+4rJvXBcx2blthfMumrvk27frxiFBLdEkH0tbGtbfB
M1um+X2pHSLotbtevdwVv05OQoB51p3Hm6m9a93Joef56lhxUHyxbAc5Kf1Jeh53UD4l8OpJ8
QXHC4D5+nAMoSk5X1Dpdx6aEGrnE23yGPPn7XE5ykIl4qkfqCJnV6VqWLTNnneC/m99o23iL
1p31tVck7eP65a3OC9q0zgvTmv0RZAU4EJbJ4PYJ4zyJNw0ZqKDpFUyGLDtJuQiKMsrfUwg/n
3tHDtiPFV20tFhRmkpal6y+Y5ZGDIqSBJtQ49/P4vFn+GhmytPNezhBLIZIprelUdxkZeNVgDgA2
yZZa6CWP/YpP8xInZReWA65fai8KYsNNlArpbP3yHKj2XFzU3S23eVONMrS1gItVQ+nGP2B05
nEXOG6u9YjDVeSVH6ZkDq+4QVLJ5st3lfbrmjCV9u7bjLJ+Vo4WRGz2LUvQdt6/Wf0onLRSAK
PrBmH9N8AdPpYMcRd2/N7SA/lPM+yNk4+jyWfySpDZnxo4r2r65lGu1i540Ra0F+H3wPrYcv7V
XQo0yGWKVyJnTq96E03ieEPgsVX9Kt0hKrdh+53CM/FDjpK/HslyF5s8+AqZTRfcfYO1qr0ssEJ
dDtK3pWyAMbtRVJMBH9R3w93vSMHzEhudwTHisgNThCrz9qymhZz1cao3JMQeOXnJHR5E9
viiUVmxS/fCQn1T9RN6zwltPCHSTs9wp5LSzhoDKGYJ2y39NG780s7uecMSyFU6ucKOhi1mP
+np4PBDzslrxy2Qm59aGMPsLmv2qGQ4tq3BpGR5ns9SbA24/7jVdv0YwA0A6RR5+Yp5N5H6
F0QCr0mUiqduHPq7XavmTX8YVGpWB/WJIpexqD0oQ3CrzwnCscy0H3huv6geMrY8Br0/xWn9
EAtlJwP9A6Td8A7xSBR5/07aghmjgwFIMDOsEwMX/FkYp5f7zvDvV7nhIFpSATs2AlnuZMei+b2
H2JDHQMS+W1/QdJ2DcDrXiuPIFE4oGJWzsZzk4qoX4GGRhH1gZyLJCcYODCNOUwrA1I+rw
4U45G7MSHSteDvcJ8qd/1HWSvMYjFm4NV/Ca91FI395IhGUcz5aQuvrb1wfaM81cjYpqXi33Xn
VlGOXgtr5tXbSMIgbIgZA9VEbo0yFa++XYBO+XZnL3UYXcknxQRxYPFKaWJJCCS7aRbpVFM
EzSyUPRev8VSrdZxeiUwQ9zWEps3uK3lFLBgYT5YNFW/bGUPsezsuHW3zpz4E61ZNzBsxQu
lxF755+BjPMgYDQ/Mqs4KozYYcJzGjaeBIA2HzpwLo+OzZPRw7Yonfylegs5SaDay+IX5OJps0k
p0tHUB8tsRvdkGo8yZcKPY4d9jNZl4Hc1kcz+K/WuXx3TnIl7Ob9EKhRTY9a/+Jh3RXFgwrHBhd
GdOThbWiMjy38KzwlyF8ZmFOdkm8meMuscsFQN3bewTcFbUqACqGCXRAjS0e7fe8CXqgUB
t8faTvuXFtHf1N3TFsyJEp89xGfPR8Ew5sTjV2n6w1H2zEN4dj9ZkkdqIwh8kaL2JD4A5IomLXEn
gkv/w3kgPE3Mk1RxWzLTrS/D/rLvLmGHc2XzsGawnQdC2kBuLgbVTNWb8F0GrbIXX84INAO7
gzw/uifMQSpU3jEv++xsZDBUMEptKGEIJcOoMgoTjqtuvOxFp3+Kn+I8IRCA0Iz+/SzrL5THCYx
HGBFmzF9OaIDWOwfShym0Dkhsk9y8Ky9YfEfY1afTayKy1BL/4iQ0gNvJaaTcUXe4LT97vkLW
tSCA0XbZdh0THQxfm4qVzrWRjnr0jPBWL4zeA//JkXUvfOyqZeEPl6Mm+rdFQvQa58Umdyc4au
5gIp2NTAGHWz6h1QOjdN+r4Ilyw8jCG6fnvCUTz5LPVvJu6njAedt1JIreV+2qhqLh8y4WZ+KNo
MwuUujEuGufXYr+Dr4agAyV4IgyOo4YtbXtlX1A3hp24d0XN2hZEFwMJ7PEJkTk0tgHNwWH77
Hnp1woTXLDWJUYFdn3umf2ct1Ph5JuEL4R0Zl8kQ1p2baTZLhPH804NF4cyz5R0BnewrNd6O
zKEcRCQdYj8xpvGQOc5WsScMPbjndRrE9R/MTQR5SMt3ODiOy5yaX+v+G4haEzSb/PlTI5IIij2
oDjcw2IXwPS+g8ny5WPmpc+AQjPjzTGU84QcfqRFANaVEfGEda++cyvKFTFcu9kXKP1vMdiB
Cgn/jq71vrTOMleAwVwdOgY1uly5p7JxVoPp0+q1XmOd8HRbTo7waxBeaT0d0w2rbazjnWXRz
5b+CZhejitmi9fIG6r0t9053Jk8QRLbHSAuokwF99xWGJjuYVCewcOIf6fD1WcLwVqdrpygNJOV
XW55vHQxOZCsw/Ir24iSdS7KFqNK3KztxI3tKHn/udn25hddtP1bbbgIr0WCMaZj5Q8aRCKjerykI
31fmNc1mBG7pKAPFL55NGdabA1gFibq1euRf+O0Uma+bYAsMYvkc2H4Pj86dog1SyZI8oUej
ClC/iESyh5yTB+ucH+z+8k6dMRjUccYeYP3rwKCBUOqp3zRMu55MSAiTqV3uEIWNpoWY6RC
miUqL319uwaw8NGy4A1aEf4M1KuN8/Q3aZYNFiQP7KOB1A2UZrH6fhcZBFAkyUh/FLroO5Rc
xp5JWNIr/jZi0NU/cG3yE6W7zlbugvGI+XNV91/yh/KKTTBCZXmpDdUjXfSjNU5OGV79TDpEIKB
eY40MOQX1a4LPATa7hMRKjwnLEkKSwrqj6kUJctkh0cUYjbDv3a/S1bN8CTZxqsHFes4MrY6z
zZ2WutUCaCuF3GI1FoAH7uqhJoI1P8CdK1op9oPhIn66BG1s9/I8E6AVp+ei2BX1wNtCpWaP/x
wDXZYGJPBEv6pzWfrINtj8p794C67LsmjCKqUwcAn013FHWIClLJozkU2vRFwhgb1s3mvmcT
moMsrfhWp1iLC03GK2OQD9AWPb2500MjFPhwfCkZH2EtaYoFIZfAexhAkqQOfWQQFHKI5vK
+JxI7bR2QmZovD4b/ebTsk69ytzsiRmyGamCPkCQKSyp+RE3bk9ypsZO4zjw226iAzhsrwuLIHL
SGjhFxNZ4h97gBK0IfmQPiI8Nvd6MF5TeUaI8ELv7HzPP45cOx1IRfAEygMOjENPJe0xbUbKU4
ENY6fwmgc2I4dJqVokfjFqp09ttyt6FQh/WnOd+d7aeQmwfSIzeFhjJLGROJyWkjM4Uirsja78tW1
PHbn5QkVNnYCbu8YowIE/N7nd/1fXwLMWcCC8IedCdgrjF73rjHiK8amSYV3A/i2MgEh6K/ccW
OhLEMf6YzN4wVpK4IzVAdc0+UxcB8DSizkKkMNhme9KCNIRLUF4cZnHTHI5LsofY8oRLkmxg

```
                    /71rL8eoJYjQ9zCuoAWNvt7M2B7uM1TYRwflWhw1ElhAHlblFmMOZukBQ+AN67LdJ9tnylCYs
                    V7eoEf4ajww+ttEMuYqNrpzjawgtKY6wR7Ob5mFlk00NgLCBss/NeZ/nYQTOoxF77jR46reHAM
                    WoIwxEcb2Eb7/RyWdHu6lVxZblEQ9HTzUa4ftZ5lxSKdMbxH9gFAcry0NzWQeFygbarjvkK5ku
                    mcsCJK51xeScJdnafyUWDuGNKks0LVSuMknfHWDRO+JXCW54A5sHy2oiwiITWnyajO8gMrt9
                    UySYyR5PBDd/pz1edmlf++TdAls7BboBUtQN1HNhtyIN6WfWzWCluK2cxMqeoh0prFYGLEkla
                    Ds05EbnAGvDHw0F4A7uPIF1VXOPLXSmw7HKSPlai2TKu5BTQ1byNGuVaeshPsuGpsw70Y
                    7nLF4DMsvBUqi2DgVf5cnBoWBWLoJQWk5TT5egklbiCpDGZ2qsBvIBvW0dmIMOMf9HTf/gdYl
                    H9JHmIndKUe+xkw9qOtF7xgDOX0WzPN0Rd7k2+MWLTxLNaakUT0KkgeoFqWTI20Uy3y1xce
                    5pDFay9cd0EFG4Ew/dwV2aGPhVgcrbjEEykH4H9TQapTJ31CUCJxGKG0BP+FoPEr/Y</xenc:
                    CipherValue></xenc:CipherData>
                </xenc:EncryptedData>
            </pkif:RenewCertificateIn>
        </soapenv:Body>
</soapenv:Envelope>
```


v. SOAP response

The next example shows a SOAP response to the RenewCertificateRequest:


```
<soapenv:Envelope xmlns:xd="http://www.w3.org/2000/09/xmldsig#"
xmlns:elem="http://danskebank.dk/PKI/PKIFactoryService/elements"
xmlns:pkif="http://danskebank.dk/PKI/PKIFactoryService" xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
        <soapenv:Header/>
        <soapenv:Body>
            <RenewCertificateOut xmlns="http://danskebank.dk/PKI/PKIFactoryService">
                <pkif:ResponseHeader xmlns="">
                    <pkif:SenderId>360817</pkif:SenderId>
                    <pkif:CustomerId>360817</pkif:CustomerId>
                    <pkif:RequestId>398</pkif:RequestId>
                    <pkif:Timestamp>2022-10-21T10:36:57Z</pkif:Timestamp>
                    <pkif:InterfaceVersion>1</pkif:InterfaceVersion>
                </pkif:ResponseHeader>

                <tns:RenewCertificateResponse xml:id="response" xmlns=""
xmlns:tns="http://danskebank.dk/PKI/PKIFactoryService/elements">
                    <tns:ReturnCode>00</tns:ReturnCode>
                    <tns:ReturnText>OK</tns:ReturnText>

                    <tns:EncryptionCert>MIIDfzCCAmegAwIBAgIHFqCSXpMUsjANBgkqhkiG9w0BAQsFADCBmjE
                    QMA4GA1UEAxMHREJHU1dESzELMAkGA1UEBhMCREsxEzARBgNVBAcTCkNvcGVuaGFn
                    ZW4xEDAOBgNVBAgTB0RIbm1hcmsxGjAYBgNVBAoTEURhbnNrZSBCYW5rIEdyb3VwMRww
                    GgYDVQQLExNEYW5za2UgQmFuayBEZW5tYXJrMRgwFgYDVQQFEw82MTEyNjIyODc3MzA
                    xMDEwHhcNMTAxMDIxMTAzNjU3WhcNMTIxMDIxMTAzNjU3WjCBlDEmMCQGA1UEAxMdSE
                    FOUyBQLkhBTINFTiBPRyBKWVRURSBIQU5TRU4xCzAJBgNVBAYTAkRLMScwJQYDVQQK
                    Ex5FU0JKRVJHIFhYWFhYWFhYWFhYWCAgICAgICAgIC4xNDAyBgNVBAUTK1NFLU5SL0R
                    BQkE6MDAxNDk4MTkxOC1BR1I6NjYwNjg0LVVTUjozNjA4MTcwgZ8wDQYJKoZIhvcNAQEBB
                    QADgY0AMIGJAoGBAK9Z2GNUK5Zda7gfxapQka6+Z77qXSDft8Be6vdebbxEKHO+G57um/C
                    G2xkTIc3c5r0OEsxX1RALjDLkx3smz6ubYe2A60zYsyisTgZ6hNimLI/V9tgNSoePbKCSRYOpD2
                    /TFIImQYjJb7W+ytwh34MOVmoFpy5fC2/nHLymWZfvAgMBAAGjUjBQMB8GA1UdIwQYMBaA
                    FA7mcPtCSqVAKvGKB0mdxNijbS0jMB0GA1UdDgQWBBSz1omIL7EVQOwKwDA1OrfacnMbT
                    TAOBgNVHQ8BAf8EBAMCBDAwDQYJKoZIhvcNAQELBQADggEBAIbgwk+wdC/q01ckSOZKH
                    xKIS3wKwu+TYcTZIj6mfSYmh2/mE5DEPQM6XZTgpDVgGggwhnAfk6hp3GJ60wHLR4tG6jhH
                    +RcJ6kKo1Ib6fVgsqQATJRhqIkW400kBtTd+33xg3cQbwiXU5Sbu5aOckReN/ELigyxwtWcl+TW
                    fC7KwWvz+Y7zLg3wE1u3xiQjGEvmdcqIrzI+7UEr4KqSuwIHEb2VGpNF20A4HaGtwdtvxZGwC
                    6IM2gnN7dFZFTjr/DSMduo57ZoXomfw6WTQgGoQdEVrB4nHfJVpEjh5/I1s38QyPqR7dfnQaQy
                    WC1qiWLBJp/qpXUTS1t3DNN4SCg8w=</tns:EncryptionCert>

                    <tns:SigningCert>MIIDfzCCAmegAwIBAgIHFqCSXpMUsTANBgkqhkiG9w0BAQsFADCBmjEQ
                    MA4GA1UEAxMHREJHU1dESzELMAkGA1UEBhMCREsxEzARBgNVBAcTCkNvcGVuaGFnZ
                    W4xEDAOBgNVBAgTB0RIbm1hcmsxGjAYBgNVBAoTEURhbnNrZSBCYW5rIEdyb3VwMRww
                    GgYDVQQLExNEYW5za2UgQmFuayBEZW5tYXJrMRgwFgYDVQQFEw82MTEyNjIyODc3MzA
                    xMDEwHhcNMTAxMDIxMTAzNjU3WhcNMTIxMDIxMTAzNjU3WjCBlDEmMCQGA1UEAxMdSE
                    FOUyBQLkhBTINFTiBPRyBKWVRURSBIQU5TRU4xCzAJBgNVBAYTAkRLMScwJQYDVQQK
                    Ex5FU0JKRVJHIFhYWFhYWFhYWFhYWCAgICAgICAgIC4xNDAyBgNVBAUTK1NFLU5SL0R
```

```
BQkE6MDAxNDk4MTkxOC1BR1I6NjYwNjg0LVVTUjozNjA4MTcwgZ8wDQYJKoZIhvcNAQEBB
QADgY0AMIGJAoGBAJ+H5qMS+wVLWQHI+yNF474SbIWCUYOIIGHu+jJrZpJu8YxCb9oBi6B
OWICGiSWrhp3BTCqN/zqCifdI1QwMGYOkmI+XF57EEd6w6OvfaHHM7y7ru1LmjXF8N7HeBrX
3aD73ikAmyqafcVzwsnRafbMk+1gJNi9uvHrnnVAuIPd/AgMBAAGjUjBQMB8GA1UdIwQYMBaA
FA7mcPtCSqVAKvGKB0mdxNijbS0jMB0GA1UdDgQWBBS38hNojcGbpBiymGe1aurA4farvzA
OBgNVHQ8BAf8EBAMCBsAwDQYJKoZIhvcNAQELBQADggEBAKEAKXRzHtRtHuBCRTwJiO
KEfUwePD4H2FSSRFcLlaCGQpXEXXPKcN8Ek15lYSsDseLMxFqRzyMWjvb6/NeCoVSuJyDiT
vFAZBbQOY5JunD+txcWJBpYcWKhxKJdkOShhFDgfoLJUPeDJWSjHTsJNLxq1s+cJjPKPmqb/
5EVHC8oDsErsCyi81jbTcjOyuKVzlUpWzlGouHH4jBJRhSJ8T++EG2yBBKnP0ruadSRfXH2OM
iwitHNquUH4c9KyDhki+VZC/rB0U05+qcO9trDhzF6EiEcPuZMtfXnMOWhFrbszLoY5KWC/PFQ
wzrRun7yozt018yoKZVKoTJeH86yMSo=</tns:SigningCert>

<tns:CACert>MIIEFzCCAv+gAwIBAgIFAc+WvjUwDQYJKoZIhvcNAQELBQAwgZgxEDAOBgNV
BAMTB0RCR1JPT1QxCzAJBgNVBAYTAkRLMRMwEQYDVQQHEwpDb3BlbmhhZ2VuMRAwD
gYDVQQIEwdEZW5tYXJrMRowGAYDVQQKExFEYW5za2UgQmFuayBHcm91cDEaMBgGA1
UECxMRRGFuc2tlIEJhbmsgR3JvdXAxGDAWBgNVBAUTDzYxMTI2MjI4MTEzMDAwMzAeFw0
xMDEwMTEwMDAwMDBaFw0yMDEwMTEwMDAwMDBaMIGaMRAwDgYDVQQDEwdEQkdT
V0RLMQswCQYDVQQGEwJESzETMBEGA1UEBxMKQ29wZW5oYWdlbjEQMA4GA1UECBM
HRGVubWFyazEaMBgGA1UEChMRRGFuc2tlIEJhbmsgR3JvdXAxHDAaBgNVBAsTE0RhbnNr
ZSBCYW5rIERlbm1hcmsxGDAWBgNVBAUTDzYxMTI2MjI4NzczMDEwMTCCASAwDQYJKoZI
hvcNAQEBBQADggENADCCAQgCggEBAOC0DrY9Q4HzHQ/BWr/RWB1GOq+90BiGN85uhW
AeGGMI2od/ahn/R7zl8+MNcMLAuVHB8LvzRrC3lxZqjNvRKv8fYsMgwdmdYtBwiqHINpXTLjQxt
LEWqEu5fiMZAi1oFE08YAkCDLUGopbkY+d2KULqxFK2bIwHi3m0jOwuZGiw4ELeChGcELsOp
CT/oJU5mR3dFHbRs3HBgWmuFwJKvwHNZeNAmrCwEUWqw1x4MjQRHRGI10TfTNqPrd7zy
TfyR0+GPy1INxHwWZroBUc9j54ONX9pV6x835WedHdo80UxImjCiIruClzsMAXYtvsC3d9xgk9o
LNFIVPRe2fTs07ECAQOjZjBkMB8GA1UdIwQYMBaAFLU6ie9kUC8x5CDLauuTRJBL35qYMB
0GA1UdDgQWBBQO5nD7QkqIQCrxigdJncTYo20tIzAOBgNVHQ8BAf8EBAMCAQYwEgYDVR
0TAQH/BAgwBgEB/wIBADANBgkqhkiG9w0BAQsFAAOCAQEAQPCp0H3g88CZZurq0kS76oz
BVRiLZ3V8S0+IYi0dmMCfTpW/qnEzMGn+NIHOvgkm5C2VaHCdbEzZPsvv4cx2YrqpsFf8x+Ts
6W2r3VVjOdve5u0Oj1CK/ONwaqUI5p4SxRCfnv6sSh6TwxhJF/zESiHXLdyWdJf+NkXsATE6Q
B4ZjgaGw1NcvhnDUvbbfUZ1zOTIf7+wUpfCNCJi0T1sFvJ88nYkVoQmVWQFS7Kwj+kwQ1ILfn
p/1xbycrt1XNxZ8SkRDAWOJARY0fS/C0o7/1t/SB2ePrY/g0U8ZWi0B6odT5isGNpLOKzhD5YGj
bKGesC9GEnhmhLoawXU7b4Wcw==</tns:CACert>
<tns:RequestId>398</tns:RequestId>

<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>h6mHrM+juFzlYC2wHx0NltdkqZs5lwGuRIrZ9A8D13g=</DigestValue>
 </Reference>
</SignedInfo>
<SignatureValue>uQs1fv2Sw6OYb3rqRXQUdwlZ26PQMdphsKmCMDD2/k9Cw8DqVxcSC9I1
U2IK2ZDbmwfmeCpfeSwRs2qNXX8ILrz7zHungsWjUIUqD9MEaTa8HJWA0khCG7gwtmG9WAl
i/ZCcQMquKiM0XuXttlg2moYTkw60kS/JR9dGDgWiHUU=</SignatureValue><KeyInfo><X509D
ata><X509Certificate>MIIDIjCCAn6gAwIBAgIFAMauIFMwDQYJKoZIhvcNAQELBQAwgcQxED
AOBgNVBAMTB0RCR1JPT1QxCzAJBgNVBAYTAkRLMRMwEQYDVQQHEwpDb3BlbmhhZ2V
uMRAwDgYDVQQIEwdEZW5tYXJrMRowGAYDVQQKExFEYW5za2UgQmFuayBHcm91cDEa
MBgGA1UECxMRRGFuc2tlIEJhbmsgR3JvdXAxGDAWBgNVBAUTDzYxMTI2MjI4MTEzMDAw
MTEJMAcGA1UEBBMAMQkwBwYDVQQqEwAxCTAHBgNVBAwTADEJMAcGA1UEERMAMB
4XDTEwMDgwNTEyMDAwMFoXDTExMTExOTEyMDAwMFowgZgxEDAOBgNVBAMTB0RCR
1NJR04xCzAJBgNVBAYTAkRLMRMwEQYDVQQHEwpDb3BlbmhhZ2VuMRAwDgYDVQQIEw
dEZW5tYXJrMRowGAYDVQQKExFEYW5za2UgQmFuayBHcm91cDEaMBgGA1UECxMRRGF
uc2tlIEJhbmsgR3JvdXAxGDAWBgNVBAUTDzYxMTI2MjI4MzMzMDAwMzCBnTANBgkqhkiG9
w0BAQEFAAOBiwAwgYcCgYEAyYTGJXapryzzumxcBXxqSdNwPPETUGg9EMI0XdovgEvyM8
2wNWrCjj2jWDdKssL2N//S2GktgkF0fqC4R38UVkM6WG0BsKoTrnJ3WIS+JIjHJdCLBAN3ZkI8
ZfkuqXqTJ+ACLLaliYFXwHCyyViy6e5RQY2XKG6jJVh699J2pZUCAQOjPzA9MBAGA1UdDgQ
JBAdEQkdTSUdOMBkGA1UdIwQSMBCADsLGxMP29sL29/X39cXEMA4GA1UdDwEB/wQEAw
```

```
IGwDANBgkqhkiG9w0BAQsFAAOCAQEAK+ZwVkaEo6FdLdn+xDNT14f8raNoo/xzQ0Qj2Z9W
XTvsWIelMNMPlBMROsUKpKr4rx0LG4gSHp5YUtT/rBcp4qxC60qol1Ezl2OezcqUu3gq9M9gdM
AYgot79FjqG6rp+bsr77stwaJgl+I5qFEz+PWDhLRgohWKP+r/ZSoj0n6X6gTLa/cGCRr2pPK2KJ
9i4hd85tySuUbMaBkXE1mywCBJe6JyhfUQpQFLmJ7kSFaBQLDdQZP2iQn2KkREz60d1c39ds
TzSGvHfusD8gVdbNS4+sPXxM4SocKXSTXwaN131tlgbzDANnsmq4TGQmcxRadaLwYhg7ZE
g45lJYYwrw==</X509Certificate><X509IssuerSerial><X509IssuerName>postalCode=, title=,
GN=, SN=, serialNumber=611262281130001, OU=Danske Bank Group, O=Danske Bank
Group, ST=Denmark, L=Copenhagen, C=DK,
CN=DBGROOT</X509IssuerName><X509SerialNumber>3333330003</X509SerialNumber></
X509IssuerSerial></X509Data></KeyInfo>
</Signature>

</tns:RenewCertificateResponse>
</RenewCertificateOut>
</soapenv:Body>
</soapenv:Envelope>
```

Appendix C: CRL Reason codes

The CRL Reason codes accepted by the RevokeCertificate and CertificateStatus operations are a subset of the reason codes specified in X.509v3 (RFC5280).

This listing shows all of the reasoncodes listed in RFC5280. The ones that are crossed over are not supported by the PKI Factory operations:

```
RFC 5280            PKIX Certificate and CRL Profile            May 2008

CRLReason ::= ENUMERATED {
     unspecified              (0),
     keyCompromise            (1),
     cACompromise             (2),
     affiliationChanged       (3),
     superseded               (4),
     cessationOfOperation     (5),
     certificateHold          (6),
     removeFromCRL            (8),
     privilegeWithdrawn       (9),
     aACompromise            (10) }
```

The type of the field is an integer, and the content as listed above.