Security and Message Specification for Financial Messages using Web Services

TABLE OF CONTENTS

1 VERSION HISTORY 5
2 INTRODUCTION 5
2.1 Background 6
2.2 The New Web Services Channel – Security Requirements 6
3 SECURITY SPECIFICATION 8
3.1 Usage Scenarios 8
3.2 Message Lifecycle Example 9
3.2.1 Message Generation Steps – Simplified Description 9
3.2.2 Message Lifecycle Diagram 10
3.3 Security Infrastructure 13
3.4 Role Specification 13
3.5 Levels of Security 13
3.5.1 Level 1 – Business Authentication 13
3.5.2 Level 2 – Communication Authentication 14
3.6 SOAP Message and Content Structure 14
3.7 Optional Content Encryption 17
3.8 Keys and Certificates 19
3.8.1 Introduction 19
3.9 Types of Keys and Certificates 20
3.9.1 Private Keys 20
3.9.2 Public Keys 20
3.9.3 Root Certificates 20
3.10 Cryptographic Algorithms and Key Lengths 20
3.11 Standards Used and Limitations 21
3.12 Nonrepudiation 21
3.13 Multiple business signatures 21
3.14 Transport Security 21
3.15 Terminology 21
3.16 Requirements Outside This Specification 23
3.17 Recommended Reading 23
4 MESSAGE SPECIFICATION 24
4.1 General Information 24
4.2 Operations 24
4.2.1 UploadFile 24
4.2.2 DownloadFileList 24
4.2.3 DownloadFile 24
4.2.4 DeleteFile 24
4.2.5 ConfirmFile 24
4.2.6 GetUserInfo 24
4.2.7 GetFileDetails 25
4.3 WSDL 25
4.4 XML Schemas 25
4.4.1 ApplicationRequest 25
4.4.2 ApplicationResponse 25
4.5 Using RequestHeader and ResponseHeader Elements Field by Field 25
4.5.1 RequestHeader 25
4.5.1.1 Sender Identification <SenderId> 25
4.5.1.2 RequestId <RequestId> 25
4.5.1.3 Timestamp <Timestamp> 25
4.5.1.4 Language <Language> 26
4.5.1.5 UserAgent <UserAgent> 26
4.5.1.6 ReceiverId <ReceiverId> 26
4.5.2 ResponseHeader 26
4.5.2.1 SenderId <SenderId> 26

4.5.2.2 RequestId <RequestId> 26
4.5.2.3 Timestamp <Timestamp> 26
4.5.2.4 ResponseCode <ResponseCode> 26
4.5.2.5 ResponseText <ResponseText> 27
4.5.2.6 ReceiverId <ReceiverId> 27
4.6 Using ApplicationRequest and ApplicationResponse Elements Field by Field 28
4.6.1 ApplicationRequest 28
4.6.1.1 CustomerId <CustomerId> 28
4.6.1.2 Command <Command> 28
4.6.1.3 Timestamp <Timestamp> 28
4.6.1.4 StartDate <StartDate> 28
4.6.1.5 EndDate <EndDate> 29
4.6.1.6 Status <Status> 29
4.6.1.7 ServiceId <ServiceId> 29
4.6.1.8 EnvironmentId <EnvironmentId> 29
4.6.1.9 FileReference <FileReference> 30
4.6.1.10 UserFileName <UserFileName> 30
4.6.1.11 TargetId <TargetId> 30
4.6.1.12 ExecutionSerial <ExecutionSerial> 30
4.6.1.13 Compression <Compression> 30
4.6.1.14 CompressionMethod <CompressionMethod> 31
4.6.1.15 AmountTotal <AmountTotal> 31
4.6.1.16 TransactionCount <TransactionCount> 31
4.6.1.17 SoftwareId <SoftwareId> 31
4.6.1.18 CustomerExtension <CustomerExtension> 32
4.6.1.19 FileType <FileType> 32
4.6.1.20 Content <Content> (the content in the `UploadFile`) 32
4.6.1.21 EncryptedData <EncryptedData> (the content in encrypted form) 32
4.6.1.22 Signature <Signature> (the digital signature for the ApplicationRequest) 32
4.6.2 ApplicationResponse 32
4.6.2.1 version <version attribute> 32
4.6.2.2 CustomerId <CustomerId> 33
4.6.2.3 Timestamp <Timestamp> 33
4.6.2.4 ResponseCode <ResponseCode> 33
4.6.2.5 ResponseText <ResposeText> 34
4.6.2.6 ExecutionSerial <ExecutionSerial> 34
4.6.2.7 Compressed <Compressed> 34
4.6.2.8 CompressionMethod <CompressionMethod> 34
4.6.2.9 AmountTotal <AmountTotal> 34
4.6.2.10 TransactionCount <TransactionCount> 35
4.6.2.11 CustomerExtension <CustomerExtension> 35
4.6.2.12 FileDescriptors<FileDescriptors> (in response to `DownloadFileList`) 35
4.6.2.12.1 FileDescriptor <FileDescriptor> 35
4.6.2.12.1.1 FileReference <FileReference> 35
4.6.2.12.1.2 TargetId <TargetId> 35
4.6.2.12.2 ServiceId <ServiceId> 36
4.6.2.12.3 ServiceIdOwnerName <ServiceIdOwnerName> 36
4.6.2.12.4 UserFileName <UserFileName> 36
4.6.2.12.5 ParentFileReference <ParentFileReference> 36
4.6.2.12.6 FileType <FileType> 36
4.6.2.12.7 FileTimestamp <FileTimestamp> 36
4.6.2.12.8 Status <Status> 36
4.6.2.12.9 AmountTotal <AmountTotal> 37
4.6.2.12.10 TransactionCount <TransactionCount> 37
4.6.2.12.11 LastDownloadTimestamp <LastDownloadTimestamp> 37
4.6.2.12.12 ForwardedTimestamp <ForwardedTimestamp> 37

4.6.2.12.13 Confirmable <Confirmable> 37
4.6.2.12.14 Deletable <Deletable>....38
4.6.2.12.15 SubStatusCode <SubStatusCode> 38
4.6.2.12.16 SubStatusText <SubStatUsText>....38
4.6.2.12.17 MissingTransactions <MissingTransactions> 38
4.6.2.12.18 SubType <SubType> 38
4.6.2.12.19 FeedbackFileAttributes <FeedbackFileAttributes> 38
4.6.2.12.19.1.1 FeedbackFileReference <FeedbackFileReference>....38
4.6.2.12.19.1.2 FeedbackFileType < FeedbackFileType> 38
4.6.2.12.19.1.3 FeedbackFileTypeName <FeedbackTypeName>....39
4.6.2.12.19.1.4 FeedbackFileStatus <FeedbackFileStatus>....39
4.6.2.12.19.1.5 FeedbackFileDate <FeedbackFileDate> 39
4.6.2.12.19.1.6 FeedbackTimestamp <FeedTimestamp>....39
4.6.2.12.19.1.7 FeedbackServiceId <FeedbackFileServiceId> 39
4.6.2.12.19.2 FileActionHistory <FileActioHistory> 39
4.6.2.13 UserFileTypes <UserFileTypes> (in response to `GetUserInfo`)....39
4.6.2.13.1 TargetId <TargetId> 40
4.6.2.13.2 FileType <FileType> 40
4.6.2.13.3 FileTypeName <FileTypeName>....40
4.6.2.13.4 Country <Country>....40
4.6.2.13.5 Direction <Direction>....40
4.6.2.13.6 FileTypeServices <FileTypeServices>....40
4.6.2.13.6.1 FileTypeService <FileTypeService>....40
4.6.2.13.6.1.1 ServiceId <ServiceId>....41
4.6.2.13.6.1.2 ServiceIdOwnerName <ServiceIdOwnerName> 41
4.6.2.13.6.1.3 ServiceIdType <ServiceIdType>....41
4.6.2.13.6.1.4 ServiceIdText <ServiceIdText>....41
4.6.2.14 Content <Content> 41
4.6.2.15 EncryptedData <EncryptedData> (the content in encrypted form) 41
4.6.2.16 Signature <Signature> (digital signature of the ApplicationResponse) 41
4.7 Sample Messages and Application Requests....42
4.7.1 Request SOAP Message 42
4.7.2 Response SOAP Message 43
4.7.3 UploadFile ApplicationRequest....44

1 VERSION HISTORY

Version	Date	Description of changes	Status of document
0.92	29.10.2007	Added simplified client side process description.	Preliminary
0.93	30.10.2007	Detailing work. Schema pictures added. XML examples corrected to comply with schema.	Preliminary
	November 2007	Circulated to banks for comments through Federation of Finnish Financial Services (Finanssialan Keskusliitto).	
0.94	12.12.2007	Finalised to draft status.	Draft
1.00	14.12.2007	Published.	Published
1.01	20.12.2007	Updated copyright notice.	Published
1.02	23.4.2008	Added ResponseCodes 31 and 32 for duplicate rejection reporting.	Draft update
1.03	23.9.2008	UserFileTypes.Country changed from ISO-3166-1 alpha 2 to alpha 3; modified examples; corrected typographical errors.	Draft update
1.04	21.10.2008	Added cryptographic algorithms; new message samples; removed SSL arrows from use case diagram.	Draft update
1.05	22.10.2008	Published.	Published
1.06proposal10	11.11.2019	Updated cryptographic requirements; deleted references to outdated versions of standards etc. Fixed minor typos.	Proposed for approval
1.10porposal1	4.5.2020	Updated organization names.	For approval of publication by Finanssiala Ry/Finance Finland
1.10	May 2020	Published.	Published

2 INTRODUCTION

This document is the security and message specification for a Web Services technology based standardised solution to be used for automated communications between banks and corporate customers.

This document is controlled and maintained by Nordea, OP Group and Danske Bank. This document can be utilised for implementations freely and without royalties. The latest version of this document is available from each of the aforementioned banks and in addition from Finance Finland (Finanssiala Ry).

This document describes the security solutions for this Web Services channel.

This document is one of a set of documents. The documents are:

1. Web Services Security and Message Specification.

2. Web Services Description Language (BankCorporateFileService.wsdl).
3. ApplicationRequest XML Schema.
4. ApplicationResponse XML Schema.

Banks will publish independently the following bank specific documentation:

1. PKI specification, especially the method of certificate delivery to customers.
2. Test services for customers.
3. List of supported data formats, file types, operations and services.
4. Bank specific instructions on the usage of message elements not covered in this common Security and Message Specification document.

2.1 Background

Bank customers have used bank specific or nationally standardised connectivity and security solutions to send computer generated financial data, e.g. payments, to their banks.

Web Services was chosen as the technology for the connectivity solution for automated machine-to-machine communication between banks and corporate customers.

This solution is based on international standards to enable implementation with modern software development tools and to encourage ERP and financial software vendors to implement client side adapters and communication solutions.

2.2 The New Web Services Channel – Security Requirements

The security solution of the Web Services channel security has to fulfill the following requirements.

- Be based on international standards thus enabling implementations with modern software development tools and encourage domestic and international software vendors to provide off-the-shelf adapters and software solutions.

- Allow each bank to choose the certificate issuing authorities they trust (CA – Certificate Authority).

- Use cryptographic algorithms and key lengths to allow at least a 20 year life cycle with security robust enough for financial transactions.

- Separate communication security from business data security to allow for the bank clients to outsource data generation and/or Web Services communications to separate external entities.

- Communication security limited to point-to-point connections between the bank and the communicating customer, be it the end customer or a service center or other intermediary. This means that end-to-end encryption and Web Services message level encryption are not required since SSL or VPN provide transport security.

- An optional XML Encryption mechanism for data content will be specified, but this encryption mechanism is not part of the Web Services layer of the services and is not based on Web Services Standards. XML Encryption is an XML standard and is used here for data protection, not Web Services message protection. XML Encryption and Decryption is performed by business applications at both ends, not by the Web Services client and server applications.

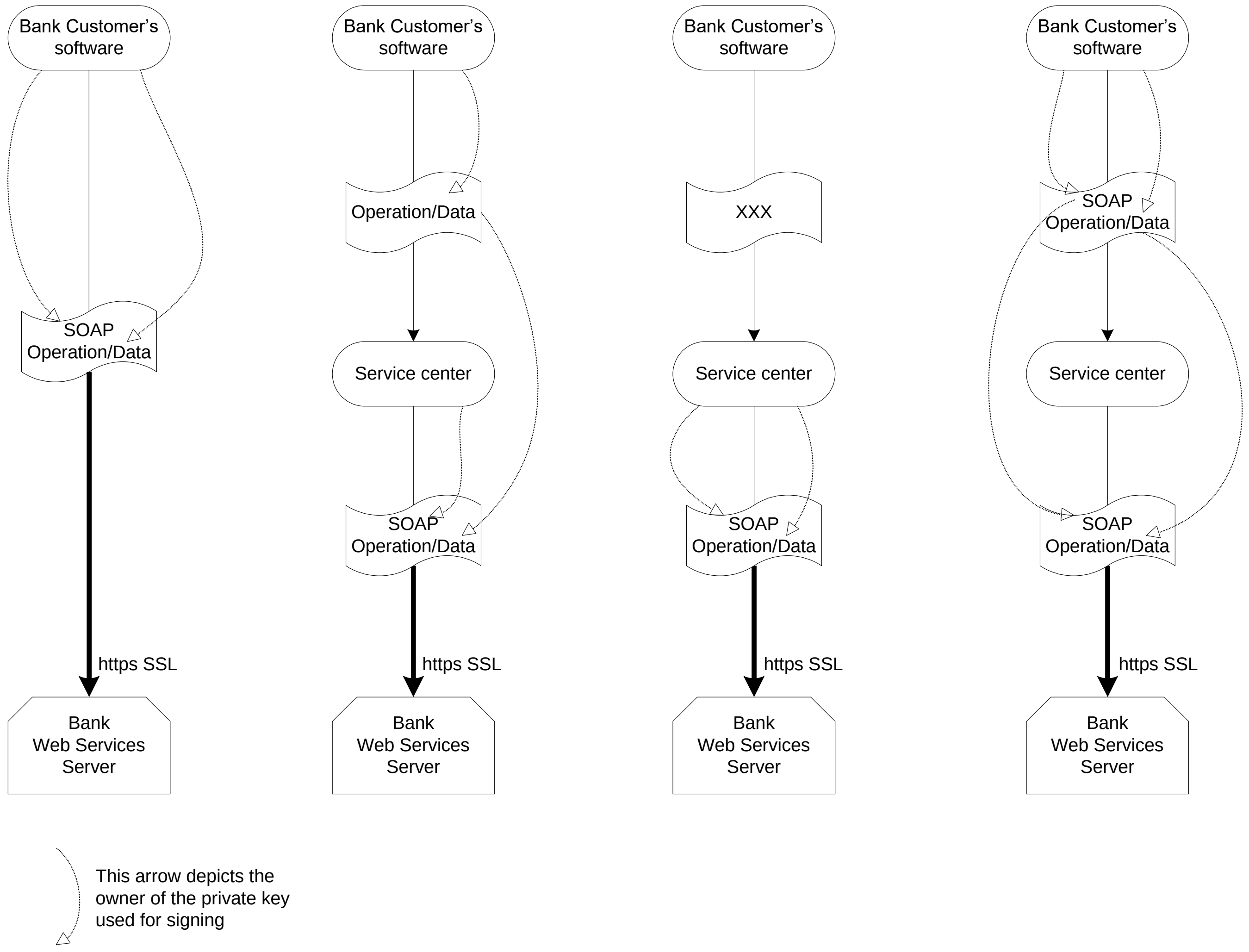
3 SECURITY SPECIFICATION

This chapter contains the security specification of the Web Services channel.

The diagrams that follow help to visualise the message structure and the different levels of authentication and security.

3.1 Usage Scenarios

Let us begin with the four basic usage scenarios.



The arrows in this diagram depict where the keys for digital signing and SSL encryption reside.

The first scenario shows the simplest setup where the Customer performs all the steps necessary:

- creates and signs the data content (payload),
- creates and signs the SOAP message,

- sends the SOAP message over HTTPS.

In the second scenario the Customer creates and signs the payload, but has outsourced the Web Services layer to another party or the Customer's internal service center. Thus the key used to sign the payload reside with the Customer, but the Sender key and SSL key are in the Service center.

The third scenario is a variation of the second one. The difference is that the Customer software does not create content in the format required by the channel. Thus the customer software cannot sign the payload and the key used for payload signing has to reside in the Service center.

In the fourth scenario the Customer generates and signs both the payload and the SOAP message. The Customer has outsourced only the HTTPS connection.

3.2 Message Lifecycle Example

3.2.1 Message Generation Steps – Simplified Description

If you think of the process at a corporate customer who is sending a payment file to the bank, the steps would be as follows:

1. Create payment file in corporate legacy format or other native format. This is called the "Payload".

2. Optional: perform compression of the Payload. Recommended algorithm is RFC1952 GZIP.

3. Perform Base64 encoding of the (optionally compressed) Payload.

4. Create an XML document called ApplicationRequest, having elements such as Content and Signature.

5. Put the Base64-coded Payload into Content element of the ApplicationRequest XML document.

6. Digitally sign (Enveloped type XML Digital Signature) the whole Application Request XML document with the Private Key of the Signing Certificate.

7. Optional: Perform XML Encryption to the Content element in the ApplicationRequest XML document to create an EncryptedData element. In the ApplicationRequest element replace the Content element by the EncryptedData element.

8. Perform Base64encoding to the signed ApplicationRequest.

9. Create a SOAP message based on the WSDL.

10. Insert the Signed and Base64-coded Application Request into SOAP message Body part.

11. Digitally sign (detached type XML Digital Signature) the whole SOAP message with the Private Key of Sender Certificate and put the signature into SOAP-header. This step is usually performed by the SOAP software based on a security configuration.

12. Send the SOAP message and wait for a response.

3.2.2 Message Lifecycle Diagram

The next diagram shows how the roles evident in the four basic scenarios interact in a service "Upload".

The Customer Legacy system generates a unit of data. This could be for example a batch of payments.

The Customer side XML Processor transform the legacy data to a format acceptable to the bank, e.g. SEPA C2B Credit Transfer Initiation XML document.

The Customer side XML Processor may optionally compress the data using the RFC1952 GZIP algorithm.

The transformed content is Base64 encoded and placed in the ApplicationRequest.Content element by the Customer side XML Processor.

The ApplicationRequest XML document is signed with the Customer's private key by the Customer side XML Processor.

The XML Processor can optionally encrypt the Content element of the ApplicationRequest document using the XML Encryption standard. This functionality is not required by this standard so please check with the service provider if they support XML Encryption. Also note that some providers may require requests to be encrypted using XML Encryption.

Only after this point does the process enter the Web Services domain.

The Customer side Web Services Client receives the XML document for further processing. It generates the RequestHeader which contains information related to the delivery of this ApplicationRequest, e.g. user ids, timestamps.

The Web Services Client generates the SOAP Message, which contains the ApplicationRequest and RequestHeader. Then the Web Services Client applies security operations to the SOAP message, in practice this means signing the SOAP Body with the Sender private key.

The SOAP Message is then delivered to the bank's Web Services Server over secure HTTP (SSL or VPN protected).

The receiving Web Services Server authenticates and validates the SOAP Message. This means identifying the Sender and checking the SOAP Message signature to verify the identification and the integrity of SOAP Message.

This ends the Web Services part of the process.

The ApplicationRequest XML document is extracted from the SOAP Message and passed to the Bank XML Processor. This step involves base64 decoding the ApplicationRequest XML document contained in the SOAP Message.

If the Content element was encrypted, it is now decrypted by the Bank XML Processor.

The Bank XML Processor authenticates the ApplicationRequest XML document by verifying the XML Digital Signature(s) and checking the validity and authority of the certificate(s). This authentication consists of identifying the Customer that generated the ApplicationRequest, verifying this identity against the signature(s) and verifying the integrity of the ApplicationRequest with the signature(s). The protocol provides for up to three ApplicationRequest Signatures (Data Generator, Verifier and Acceptor). The number of signatures depends on the type of Content and the business contract between bank and Customer.

The Bank XML Processor decodes the Base64 encoding of the Content element.

If the Content was compressed, the Bank XML Processor now decompresses the Payload.

Validation of the content means checking it against an appropriate business content XML Schema.

If the Content passed the checks described in the previous paragraph, the Bank XML Processor transforms the data to a format understood by the Bank Business Application. The transformed data is then passed to the Bank Business Application for processing.

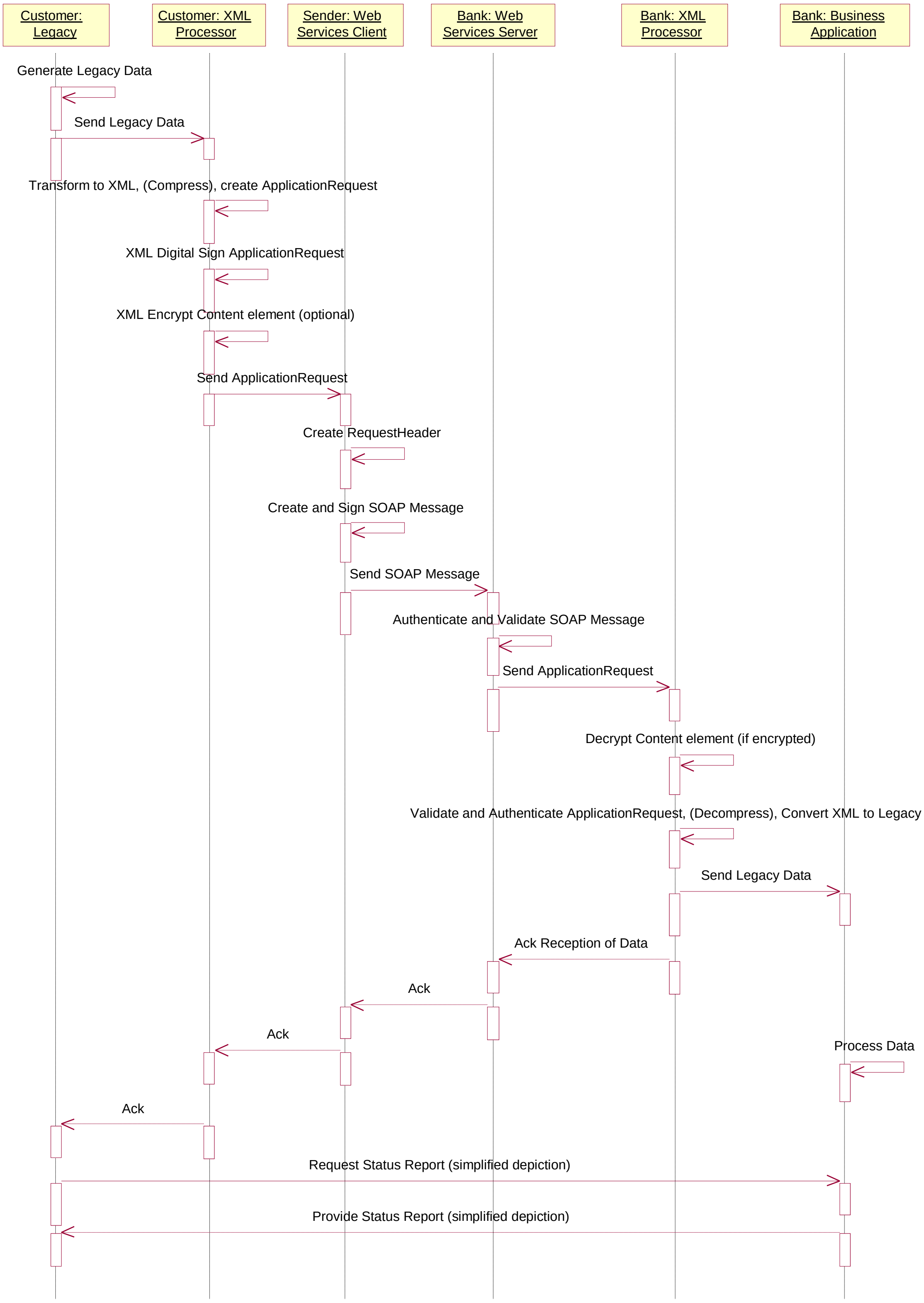
After passing the Data to be processed the Bank XML Processor generates an acknowledgment message that is passed back to the Customer Legacy through all the layers.

The processing of the Data is performed asynchronously. The Customer Legacy system has to have a mechanism to request for an application level response after an appropriate delay, e.g. two hours later. This request for application level response is shown in simplified form in this diagram. In practice it goes through the same processing as the request message.

This type of communication could be called *Request with Acknowledgment*. The immediate reply from the receiver is in the context of the delivery: message integrity, identification, authentication, content validation etc. The application level processing of the data is not performed during the HTTP connection so the application level response has to be fetched separately.

This does not preclude introducing new real time services in the future, where the Data is processed during the HTTP connection and the acknowledgment will thus be an application level response.

The Response Message sent by the Bank to the Sender contains symmetrically the same two signatures (SOAP Signature and Content Signature) as the Request Message. But in the Response Message these two signatures are made with the Bank's Private Key and are used by the Sender and Customer to verify the authenticity and integrity of the Response Message.


Customer: Legacy
Customer: XML Processor
Sender: Web Services Client
Bank: Web Services Server
Bank: XML Processor
Bank: Business Application
Generate Legacy Data
Send Legacy Data
Transform to XML, (Compress), create ApplicationRequest
XML Digital Sign ApplicationRequest
XML Encrypt Content element (optional)
Send ApplicationRequest
Create RequestHeader
Create and Sign SOAP Message
Send SOAP Message
Authenticate and Validate SOAP Message
Send ApplicationRequest
Decrypt Content element (if encrypted)
Validate and Authenticate ApplicationRequest, (Decompress), Convert XML to Legacy
Send Legacy Data
Ack Reception of Data
Ack
Ack
Process Data
Ack
Request Status Report (simplified depiction)
Provide Status Report (simplified depiction)

3.3 Security Infrastructure

The Web Services channel security and authentication is based on a PKI – Public Key Infrastructure.

Each bank can specify which Certificate Authorities they trust and thus which Certificates they accept.

The message protocol allows for both server installed "soft" certificates and personal, card based "hard" certificates. The card based certificates require a physical reader terminal and the user has to enter a PIN number every the time certificate is used.

3.4 Role Specification

The Web Services solution has two distinct client side roles:

1. Customer – this is the business customer of the bank, the party generating and processing business data to be exchanged with the bank. The Customer role can further be divided into three roles:

    a. Data Generator.

    b. Data Verifier.

    c. Data Acceptor.

    These three roles can be used only when they are required by the business contract between the bank and the Customer. Usually only one signature is required.

2. Sender – this is the intermediator handling the actual Web Services communications with the bank. The Customer can act in this role or this role can be outsourced to a third party.

3.5 Levels of Security

The security is handled at two distinctly different levels – business data and communications. This reflects the two roles Cutomer and Sender.

Separating these two levels allows using the business data security mechanisms for data delivered through different types of channels, e.g. Web Services, ftps or proprietary delivery systems.

3.5.1 Level 1 – Business Authentication

The business payload contained in the Web Services message is authenticated for business purposes.

As the business payload contains sensitive data, e.g. instructions for money transfers, this authentication is essential for establishing the authority to perform the requested operation, e.g. access bank accounts.

From a pure business perspective this level of authentication is sufficient. There is no business need to authenticate the communicating party, because the business payload in itself is authenticated.

3.5.2 Level 2 – Communication Authentication

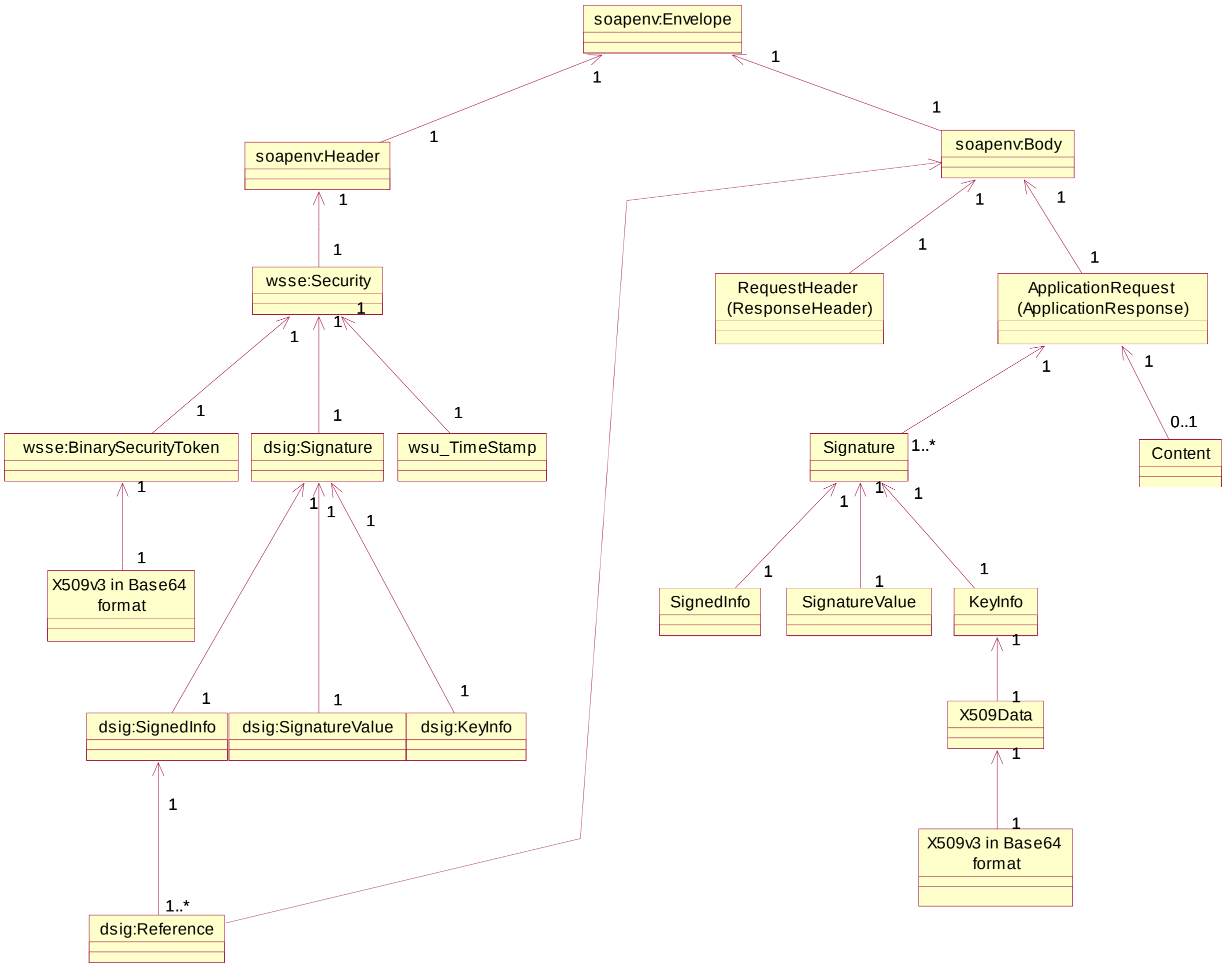
The second level of security is for communication and other technical purposes.

One of the major reasons for authenticating the communicating party is to prevent Denial-of-Service attacks. If the other party is authenticated early in the transaction, fraudulent parties can be blocked out without too much processing strain on the bank servers. The bank business applications will thus be shielded from malicious content that would result in unnecessary processing.

3.6 SOAP Message and Content Structure

The SOAP message and data content structure is described in more detail in another document: Web Services Message Structure. This document contains just a brief overview.

This diagram shows the high level structure of the SOAP message containing an ApplicationRequest element, with emphasis on the security elements.



The root element is the SOAP Envelope.

The SOAP Envelope consists of the SOAP Header and the SOAP Body.

The SOAP Header contains only elements added by the Web Services Stack based on a Security Configuration.

The SOAP Body is divided into a RequestHeader and an ApplicationRequest. ApplicationRequest may contain the element Content, which is the actual Data to be transmitted – this data is always in Base64 format and thus can be any type of data.

RequestHeader contains information pertaining to the transfer of the operation request or data, e.g. SenderId, timestamp, language code. This header is built by the Sender.

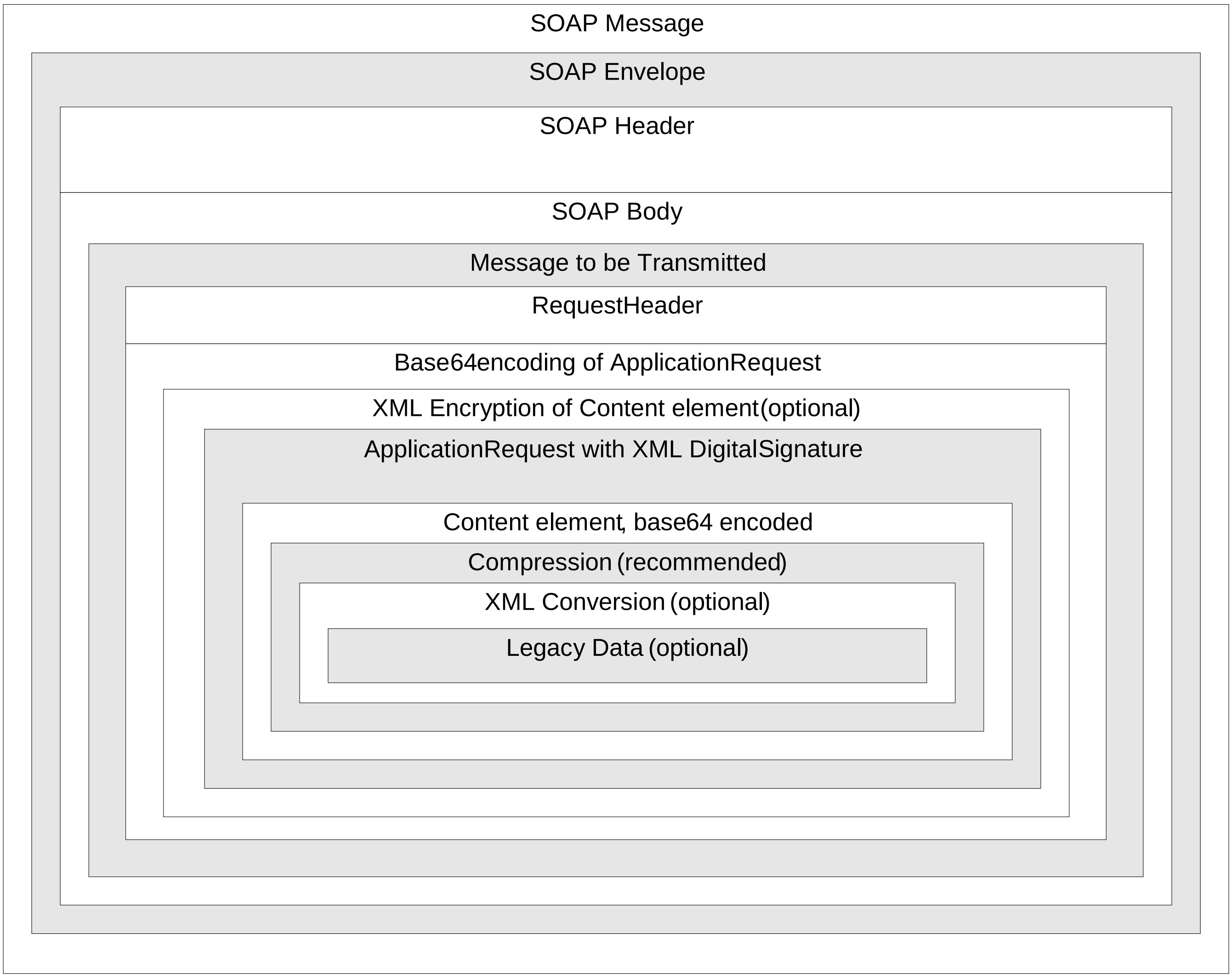
The ApplicationRequest contains information related to the operation itself or the transmitted data. This header is built and signed by the Customer. In operations where the request does not contain any data, the operation is authenticated and authorised by the Customer signature of the ApplicationRequest.

The Web Services operations are defined in the WSDL. If the structure of the messages is updated, there will be a new WSDL.

The ApplicationRequest can contain one, two or three XML Digital Signatures. These are used to authenticate the operation request or data and ensure its integrity. The three signatures correspond to the Customer roles Data Generator, Data Verifier, Data Acceptor (in Finnish: tekijä, tarkastaja, hyväksyjä). The XML signature mechanism used in Content signing is called *Enveloped Signature* – the signatures are placed inside the ApplicationRequest element which is the object of the signature.

This is in contrast to the SOAP Message signature, which is placed in the SOAP Header and contains a reference to the SOAP Body. This signature mechanism is called *Detached Signature*. The XML Digital Signature contained in the SOAP Header signs the whole SOAP Body element, thus authenticating the message and ensuring its integrity.

The layering of data in the SOAP Message is shown in the following diagram. The diagram shows a SOAP message containing an ApplicationRequest. Please note that the SOAP layers are optional – the authentication mechanisms for the data can be used even without SOAP or Web Services, e.g. if the data is delivered with ftps or some other delivery method.


SOAP Message
SOAP Envelope
SOAP Header
SOAP Body
Message to be Transmitted
RequestHeader
Base64encoding of ApplicationRequest
XML Encryption of Content element(optional)
ApplicationRequest with XML DigitalSignature
Content element, base64 encoded
Compression (recommended)
XML Conversion (optional)
Legacy Data (optional)

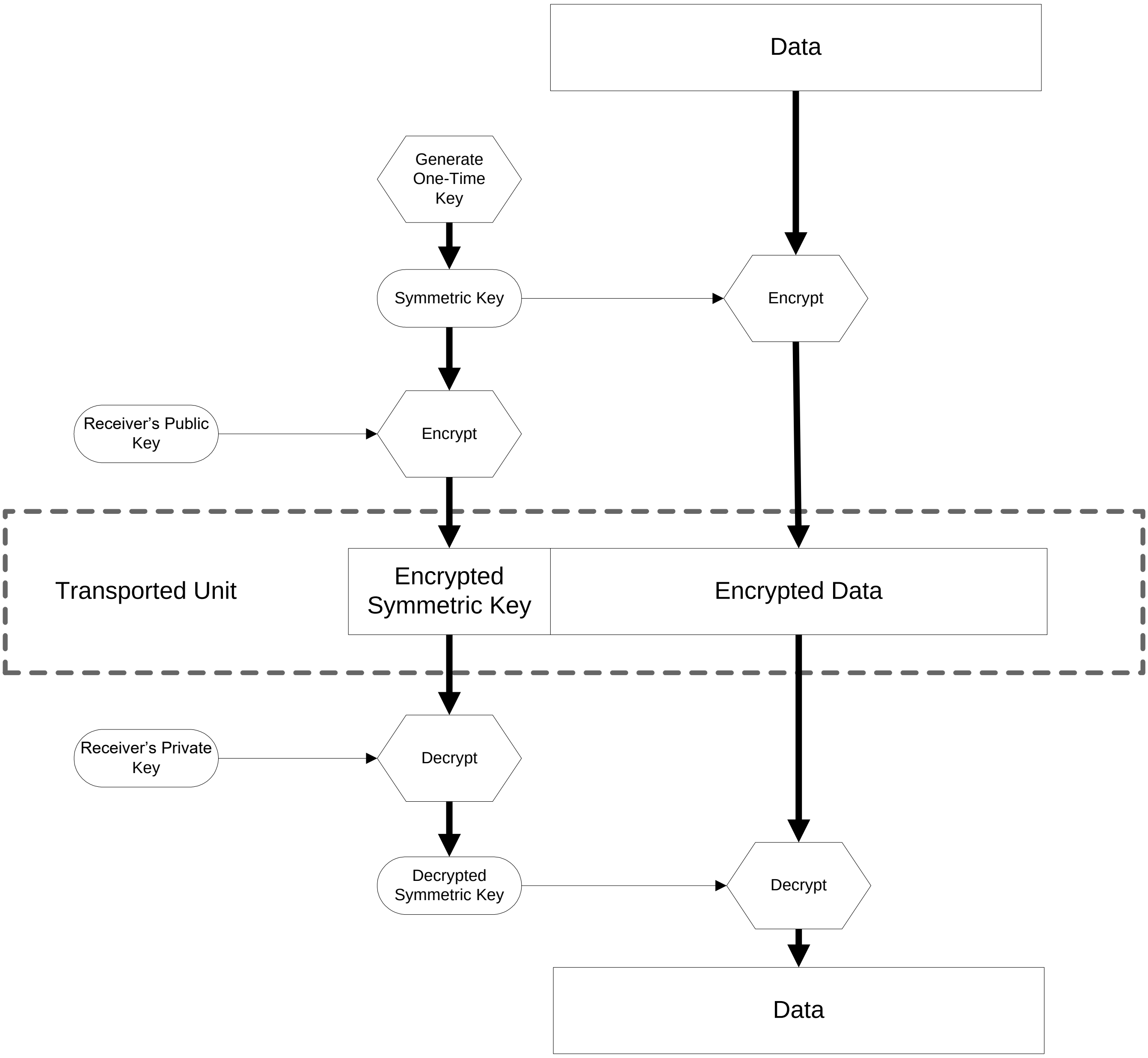
3.7 Optional Content Encryption

The Content can optionally be encrypted. The encryption mechanism used is XML Encryption. Please note that this encryption mechanism is not part of Web Services. The Web Services level security is provided at transport level by SSL or VPN.

XML Encryption is performed with a symmetric algorithm. The encryption is done with a generated, one time symmetric key. This symmetric key is encrypted with the receiver's public key and the resulting encrypted symmetric key is attached to the encrypted data.

The receiver first decrypts the symmetric key with its private key. Then it uses the decrypted symmetric key to decrypt the actual data content.

This process is visualised in the following diagram.

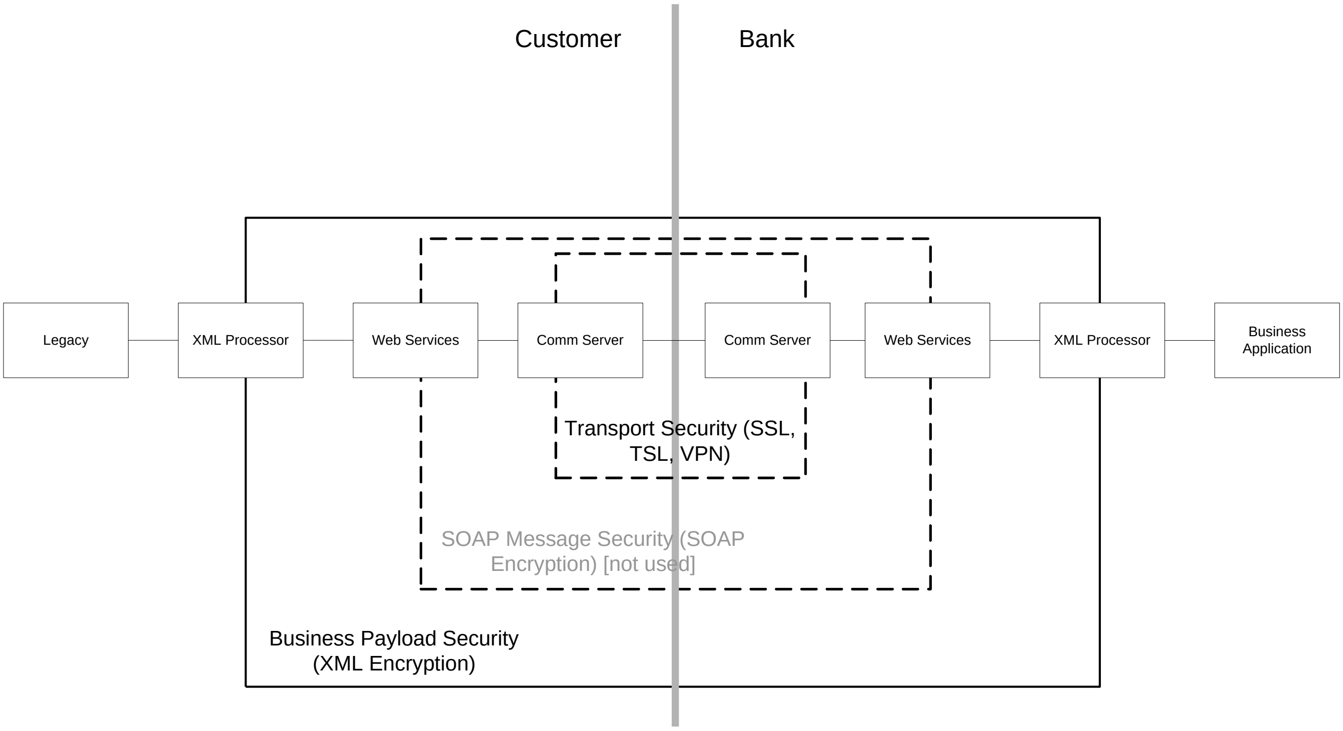


The three potential encryption domains are visualised in the following diagram.

Transport Security applies only to the delivery which is performed between two communications servers.

SOAP Message Security applies only to the Web Services domain. This is closely related to the Transport domain and is thus considered redundant in this service.

Business Payload Security is applied from one business application to another business application. This security is useful because it works the same regardless of the delivery method. This is the optional XML Encryption specified in this document. Please note that it is not dependent on Web Services. Also note that Business Payload encryption means that Web Services and lower transport layers do not have any access to the contents of the Business Payload.


Customer
Bank
Legacy
XML Processor
Web Services
Comm Server
Comm Server
Web Services
XML Processor
Business Application
Transport Security (SSL, TSL, VPN)
SOAP Message Security (SOAP Encryption) [not used]
Business Payload Security (XML Encryption)

3.8 Keys and Certificates

3.8.1 Introduction

The ApplicationRequest XML document, i.e the actual business data, the payload and its header fields, is signed with a private key. This key is called the Customer Key. This signature is used to authenticate the bank business Customer and ensure integrity of the Content (i.e. the Content has not been modified after it has been signed).

The SOAP message is signed with another private key, the Sender Key. This signature is used to authenticate the Sender and ensure integrity of the message.

The HTTP SSL connection is established with its own keys which are separate from the Customer Key and Sender Key.

The Customer and Sender keys can be the same key if the Customer does not outsource parts of the process.

It is essential that all three keys *can* be different so the business process and communications security issues are decoupled. This means for example the possibility for customers to outsource the generation and delivery of SOAP Messages to a third party but retain the data generation in-house.

Please note that the signatures are used for two logically separate purposes:

1. To authenticate the party which generated the signed data.
2. To ensure that the signed data has not been modified in transit.

The signing of the Content can be performed in the business application which generates the Content. This signing can happen long before the SOAP Envelope is generated and in a different environment.

The signing of the SOAP Message that is placed in the SOAP Header is performed in the Web Services Client or Server application at the time the SOAP Message is generated.

3.9 Types of Keys and Certificates

Each party has in its possession three types of keys. Some of these key are delivered to the parties as certificates (X.509v3) either beforehand or in a received SOAP Message.

3.9.1 Private Keys

Private keys are used for signing and decryption.

Private keys have to be protected against unauthorised use and protected against unauthorised modifications.

Private keys are strictly not to be shared with anyone. Private keys have to be stored very securely, since in wrong hands they can be used to sign fraudulent Content.

3.9.2 Public Keys

Public keys are used for verifying signatures and performing encryption.

Public keys have to be protected against unauthorised modifications.

Each party already has in its possession or receives in a SOAP message the public key(s) of the other party. When the key(s) are delivered in a SOAP message, they are in X.509v3 certificate format.

3.9.3 Root Certificates

Root certificates are used to verify and authenticate the certificates containing public keys of other parties.

Root certificates are issued by trusted Certificate Authorities. The banks have the authority to decide which Certificate Authorities they trust, in effect deciding which certificates are accepted by the bank.

Root certificates must be well protected against unauthorised modifications.

By changing a root certificate fraudulently a malicious party could make the security mechanism trust false certificates.

3.10 Cryptographic Algorithms and Key Lengths

Use of cryptographic algorithms should follow best practice recommendations. This includes hash functions, digital signature algorithms, and encryption algorithms.

3.11 Standards Used and Limitations

The solution is based on the following Web Services standards:

WS-I Basic Profile
WS-I Basic Security ProfileWS Security X.509v3 Certificate Token, OASIS Standard
WS Security
SOAP
WSDL
HTTPS
SSL/TLS

Best practice recommendations should be followed regarding which versions to use and/or support.

3.12 Nonrepudiation

The nonrepudiation chain has to be unbroken. In addition to verifying the Customer(s) and Sender's digital signatures, the recipient must also verify the validity of the certificate used. This includes verifying the signatures on all certificates in the certificate chain (i.e., the issuer's digital signature on the subject's certificate) as well as ensuring that none of the certificates have been revoked.

3.13 Multiple business signatures

There are two ways to enable multiple signatures of the ApplicationRequest XML document. Which one is used is to be determined by the bank offering support for multiple signatures.

One method is to use to do multiple Enveloped XML Digital Signatures. Each signature will sign all the previous signatures along with the data of the request.

```
ApplicationRequest
   Signature
   Signature
   Signature
```

3.14 Transport Security

Transport security can be provided by TLS or VPN.

Other transport protocols can also be used, but they are out of scope of this specification and are left to be agreed upon by the bank and its customers.

3.15 Terminology

Business Application	a software application which does not necessarily support XML and Content security mechanisms specified for this Web Services channel; ; in this document this term is used mainly on the Bank side
CA	see Certificate Authority

Certificate Authority	an organisation which authenticates certificates of other parties by signing them with the CA's private key
Certificate	a file containing a public key with administrative information and signed by a Certificate Authority (with the Certificate Authority's private key)
Content	bank service data in a format specified by this Web Services channel, e.g. Base64 encoded XML document; also contains signature(s) of the content (Enveloped Signature)
ContentHeader	an XML element containing information related to a bank service request, e.g. Customer userid, Timestamp, Data encoding.
Customer	bank business customer that creates and receives bank service data; has an agreement with a bank to use certain bank services
Data	bank service data in a native or legacy format, e.g. LMP or LUM
Intermediary	party that handles bank service request on behalf of a Customer; a future option, not used in this version of the specifications
Legacy	a software application which does not necessarily support XML and Content security mechanisms specified for this Web Services channel; in this document this term is used mainly on the Customer side
PKI	see Public Key Infrastructure
Public Key Infrastructure	an infrastructure where parties use private and public keys to perform encryption, decryption, signing using asymmetric algorithms
Private Key	a PKI asymmetric key which is kept secret
Public Key	a PKI asymmetric key which can be published to other parties
Sender	party that communicates with a bank using this Web Services channel; has an agreement with a bank to use the Web Services channel
Web Services Client	the client (Customer) side Web Services application that posts Web Services service requests to a bank
Web Services Server	the server (Bank) side Web Services application that received and processes Web Services service requests from Customers
WS Interoperability	a guideline that specifies how to apply and implement different WS standards; applications built according to WS Interoperability rules can cooperate because they are implemented the same way
WSDL	an XML document describing a set of Web Services services, their operations and parameters
X.509	an XML document schema for a certificate
XML Encryption	a standard for encrypting XML documents

XML Processor	a software application that handles data transformations between Legacy and XML (Data and Content) and security operations related to the Content

3.16 Requirements Outside This Specification

The generation, distribution, administration and use of PKI key pairs and certificates is outside the scope of this specification.

3.17 Recommended Reading

Bishop, Matt. Computer Security – Art and Science. Addison Wesley 2003. ISBN 0-201-44099-7.

Cauldwell Patrick etc. Professional XML Web Services. Wrox Press 2001. ISBN 1-861005-09-1.

Panko, Raymond R. Corporate Computer and Network Security. Prentice Hall 2004. ISBN 0-13-121191-9.

Web Services Interoperability WS-I http://www.ws-i.org/.

4 MESSAGE SPECIFICATION

4.1 General Information

Every SOAP message contains the operation and data, parameters and filters in an XML element called ApplicationRequest or ApplicationResponse.

Every ApplicationRequest element must be signed by the customer and this signature is of type XML Signature Enveloped and thus is contained within the ApplicationRequest element itself. This signature is used by the bank to authenticate the operation and to verify the integrity of the operation request and data.

Every ApplicationResponse element is signed by the bank and this signature is of type XML Signature Enveloped and thus is contained within the ApplicationResponse element itself. This signature is used by the customer to authenticate the response and to verify the integrity of the response and data.

4.2 Operations

4.2.1 UploadFile

Used to upload data to the bank.

One file per message.

4.2.2 DownloadFileList

Used to get a list of files in the bank – both files uploaded by the customer and files waiting to be downloaded by the customer.

4.2.3 DownloadFile

Used to download one file from the bank. This operation requires that the customer has used the `DownloadFileList` operation or other means to obtain the unique file reference. This file reference is used to identify the exact file to be downloaded.

4.2.4 DeleteFile

Delete a file. Can be used for example to cancel a file uploaded by the customer, but only if the file has not been processed yet. If the file is already forwarded to processing, it cannot be deleted anymore, i.e. the bank system will refuse to delete it.

4.2.5 ConfirmFile

4.2.6 GetUserInfo

Used to get information on file types which are accessible to the user.

4.2.7 GetFileDetails

Used to get detailed information on an individual file.

4.3 WSDL

WSDL file is provided separately.

4.4 XML Schemas

XML schema files are provided separately. This section contains graphical representations of the two XML schemas.

4.4.1 ApplicationRequest

4.4.2 ApplicationResponse

4.5 Using RequestHeader and ResponseHeader Elements Field by Field

4.5.1 RequestHeader

4.5.1.1 Sender Identification <SenderId>

Presence: [1..1]
Definition: The unique identification of the sender of this request message. The message sender can be a 3rd party service bureau. This indentification is issued and managed by the receiver of this message (the bank). The SenderId identity is authenticated by the digital signature in the SOAP Header.
Data Type: Max35Text

4.5.1.2 RequestId <RequestId>

Presence: [1..1]
Definition: The unique identification for this request.
Data Type: Max35Text
Rule: This unique ID is copied to the response header. This value must be unique for three months.

4.5.1.3 Timestamp <Timestamp>

Presence: [1..1]
Definition: Time and date when the request was sent.
Data Type: ISODateTime, if no time zone specified, UTC time zone assumed

4.5.1.4 Language <Language>

Presence: [0..1]
Definition: Language attribute. Used to request language version for certain information in display (human readable) format.
Data Type: Max16Text

If Code, one of the following codes must be used:

Code	Name
EN	Enlish
SV	Swedish
FI	Finnish

4.5.1.5 UserAgent <UserAgent>

Presence: [1..1]
Definition: The name and version of the software which was used to send this request.
Data Type: Max35Text

4.5.1.6 ReceiverId <ReceiverId>

Presence: [1..1]
Definition: Identification of the receiver of this request message
Data Type: BIC for the bank

4.5.2 ResponseHeader

4.5.2.1 SenderId <SenderId>

Presence: [1..1]
Definition: The unique identification of the sender of the original request message for this response (the receiver of this response).
Data Type: Max35Text

4.5.2.2 RequestId <RequestId>

Presence: [1..1]
Definition: The unique identification copied from the original request for this response.
Data Type: Max35Text

4.5.2.3 Timestamp <Timestamp>

Presence: [1..1]
Definition: Time and date when the response was sent.
Data Type: ISODateTime

4.5.2.4 ResponseCode <ResponseCode>

Presence: [1..1]
Definition: This code is used to indicate the file delivery (send – receive) condition.
Data Type: Code, Max16Text

If Code One of the following codes must be used:

Code	Name	Remarks
00	OK.	

01	Pending.	not used
02	SOAP signature error.	signature verification failed
03	SOAP signature error.	certificate not valid for this id
04	SOAP signature error.	certificate not valid
05	Operation unknown.	
06	Operation is restricted.	
07	SenderID not found.	
08	SenderID locked.	
09	Contract locked.	
10	SenderID outdated	
11	Contract outdated	
12	Schemavalidation failed.	
13	CustomerID not found.	
14	CustomerID locked.	
15	CustomerID outdated.	
16	Product contract outdated.	
17	Product contract locked.	
18	Content digital signature not valid.	
19	Content certificate not valid.	
20	Content type not valid.	
21	Deflate error.	
22	Decrypt error.	
23	Content processing error.	
24	Content not found.	
25	Content not allowed.	
26	Technical error.	
27	Cannot be deleted.	
28	[not used]	not used
29	Invalid parameters.	
30	Authentication failed	
31	Duplicate message rejected.	SOAP.Body.RequestHeader.SenderId + SOAP.Body.ReqhestHeader.RequestId
32	Duplicate ApplicationRequest rejected.	ApplicationRequest.CustomerId + ApplicationRequest.Timestamp
33	TargetID not found	
34	Contract not found	
35	Authorization failed	
36	Technical error, contact bank helpdesk	

4.5.2.5 ResponseText <ResponseText>

Presence: [0..1]
Definition: The textual explanation of the condition.
Data Type: Max512Text
Rule: See the response code list

4.5.2.6 ReceiverId <ReceiverId>

Presence: [1..1]
Definition: Identification of the receiver of the original request message for this response (the sender of this response)
Data Type: BIC for the bank

4.6 Using ApplicationRequest and ApplicationResponse Elements Field by Field

4.6.1 ApplicationRequest

This section describes the use of XML element ApplicationRequest field by field.

4.6.1.1 CustomerId <CustomerId>

Presence: [1..1]
Definition: Code used by the bank to identify the customer who originated this request. This code is bank specific, i.e. each bank issues and manages its own **CustomerIds**.
When signing the ApplicationRequest element, the certificate used to verify the **Signature** must be associated with the **CustomerId** given in this field.
CustomerId identifies the customer, the **Signature** authenticates the identity of the customer.
This element is always mandatory in all operations.
Data Type: Max16Text (min 1, max16)
Rule: If rejected, “mandatory field missing” field=XXX

4.6.1.2 Command <Command>

Presence: [0..1]
Definition: This element specifies the requested operation.
The values are not case sensitive.
This element is optional if the bank can determine the operation by other means. For example in the SOAP message the name of a higher level XML element can already specify the operation. In such a case, the Command element can be left out, but if it is included in the request, its content must match the operation specified by other means.
Data Type: Code, Max32Text
Rule: If the command is specified by two separate means and they do not specify the same command, the request is rejected (Status = Command Mismatch)

If Code, one of the following codes must be used:

Code	Name
UploadFile	send a file
DownLoadFileList	request a list and detailed information of downloadable files
DownloadFile	download a file
DeleteFile	delete a file
ConfirmFile	confirm a file
GetUserInfo	request

4.6.1.3 Timestamp <Timestamp>

Presence: [1..1]
Definition: Time and date when the Application Request Header was created
Data Type: ISODateTime

4.6.1.4 StartDate <StartDate>

Presence: [0..1]
Definition: When requesting data from the bank, e.g. with the DownloadFileList operation, this element can be used to specify filtering criteria.

This element contains a date which specifies the starting point of the time filter, inclusive. If this element is not present, but EndDate is given, it means the filtering criteria does not have a starting point.
Data Type: ISODate

4.6.1.5 EndDate <EndDate>

Presence: [0..1]
Definition: When requesting data from the bank, e.g. with the DownloadFileList operation, this element can be used to specify filtering criteria.
This element contains a date which specifies the ending point of the time filter, inclusive. If this element is not present, but StartDate is given, it means the filtering criteria does not have an ending point.
Data Type: ISODate

4.6.1.6 Status <Status>

Presence: [0..1]
Definition: When requesting data from the bank, e.g. with the DownloadFileList operation, this element can be used to specify filtering criteria.
One status can be specified in this element and this status is used to filter the requested data. For example only a list of files with status "NEW" can be fetched.
Data Type: Code, Max10Text
Rule: If omitted, default value is all files, Code = “ALL”

If Code, one of the following codes must be used:

Code	Name
NEW	Give me a list those files that haven’t been downloaded, yet
DLD	Give me a list of those files that have already been downloaded
ALL	Give me a list of both new and already downloaded files

4.6.1.7 ServiceId <ServiceId>

Presence: [0..1]
Definition: Additional identification information of the Customer, for example a Contract Number, Account Number or similar. This element is used if the CustomerId alone does not give identification that is specific enough to process the request.
Data Type: Max256Text

4.6.1.8 EnvironmentId <EnvironmentId>

Presence: [1..1]
Definition: This field specifies which environment the request is meant for.
The values are not case sensitive.
This element must be agree with the URL the request was sent to. For example if this element says "Production", but the request was sent to a test URL, the bank will reject the request. The customer can use this element to add a level of redundancy which helps to catch situations when a wrong URL is used.
Data Type: Code, Max16Text
Rule: In case of URL and code mismach the following Response code is given “Environment mismatch”

If Code, one of the following codes must be used:

Code	Name
PRODUCTION	Production environment
TEST	Testing environment

4.6.1.9 FileReference <FileReference>

Presence: [0..1]
Definition: Unique identification of the file that is the target of the operation.
This element is used in operations `DownloadFile, DeleteFil`e and `ConfirmFile` to specify which file is to be operated upon.
The customer must have obtained the FileReference value beforehand, e.g. using the DownloadFileList or UploadFile operations.
The customer never generates the FileReference. This value is generated in the bank. It is comparable to a file system File Handle.
Data Type: Max16Text
Rule: This element mandatory in operations `DownloadFile, DeleteFile` and `ConfirmFile`, where there is a need to specify one specific file as the target of the operation. This element is ignored in other operations.

4.6.1.10 UserFileName <UserFileName>

Presence: [0..1]
Definition: A name given to the file by the customer.
The value given in this element in the `UploadFile` operation is stored in the bank and shown in various listings to help the customer identify the file.
Please note that the real identification of a file is the FileReference. The UserFileName field is just comment type information and is not used by bank systems.
Data Type: Max80Text
Rule: This element is mandatory in the operation `UploadFile` and ignored in all other operations. If missing, request will be rejected, responsecode = "No File Name"

4.6.1.11 TargetId <TargetId>

Presence: [1..1]
Definition: The logical folder name where the file(s) of the customer are stored in the bank. A user can have access to several folders.
A customer may want to give their users different views of files and assets that are included in the customer agreement. That can be achieved by organizing file types and assets associated to those file types into separate folders.
Data Type: Max80Text
Rule: Optional for information requests, if omitted the response will cover all files that the user has access to.

4.6.1.12 ExecutionSerial <ExecutionSerial>

Presence: [0..1]
Definition: An identifier given the customer to identify this particular request. The bank does not enforce the uniqueness of this identifier – the value is used only by the customer.
This element could be used for example to uniquely identify all `ConfirmFile` operations.
This element is optional. Using ISO timestamp is recommended.
Data Type: Max32Text

4.6.1.13 Compression <Compression>

Presence: [0..1]
Definition: Compression indicator for the content and compression request for the responses.
Data Type: Boolean

Rule: If this element is present and the content is string *tru*e (case-sensitive) it means that the Content is compressed.
If this element is present and the content is string *false* (case-sensitive) it means that the Content is NOT compressed.

Code	Name
true	Payload in Content is compressed
false	Payload in Content is not compressed

4.6.1.14 CompressionMethod <CompressionMethod>

Presence: [0..1]
Definition : Name of the compression algorithm
DataType: Max35Text
Rule: Currently, the only compression algorithm supported is RFC1952 GZIP. Other algorithms may be used but are not covered by this specification. Compression is performed on the original raw data, i.e. before base64 encoding and before placing the data in the Content element.
The following is a table of the content in the element and the corresponding algorithms

Code	Name
GZIP	gzip (RFC1952) http://tools.ietf.org/html/rfc1952

4.6.1.15 AmountTotal <AmountTotal>

Presence: [0..1]
Definition: Total sum of amounts in the file**.**
If the data contained in this request has monetary values, the customer can calculate the total amount of these values and place it in this field. If this element is present, the bank can compare the values in the data against the value in this element and reject the request if they do not match. The use of this check is to be agreed between the customer and the bank.
This element can also be used in file listings and reports as comment type information to help identify data files. It is easier for the customer to say "file sent last week with total amount around 2.000 euros", instead of "file with FileReference 192830384938". It is up to the bank to decide if it takes the Amount information from this element in the request or if it calculates it from the data file.
Data Type: Double

4.6.1.16 TransactionCount <TransactionCount>

Presence: [0..1]
Definition: The same use as element **AmountTota**l, but contains the total number of transactions in the data. What "transaction" means varies with type of data, e.g. in C2B pain.001.001.02 payment data TransactionCount is the number of <CdtTrfTxInf> elements.
Data Type: Long

4.6.1.17 SoftwareId <SoftwareId>

Presence: [1..1]
Definition: This element contains the name and version of the client side software that generated the request. It is used for customer support purposes.
Data Type: Max80Text

4.6.1.18 CustomerExtension <CustomerExtension>

Presence: [0..1]
Definition: Customer, bank, country or region specific elements not already contained in the schema. This element allows adding new element without changing the ApplicationRequest schema. Both customer and bank must agree on the structure of this element.
Data Type: Complex AnyType

4.6.1.19 FileType <FileType>

Presence: [0..1]
Definition: Specified the type of file in the request. Can also be used as a filter in the operation `DownloadFileList`.
The values accepted in this element are agreed upon between the customer and the bank.
New file types will be added, and they will not affect the schema. An appendix will be provided listing commonly used FileTypes.
Data Type: Max40Text
Rule: For ISO messages, the ISO name must be used. This element is mandatory in operation `UploadFile`, optional in `DownloadFileList`, ignored in other operations.

4.6.1.20 Content <Content> (the content in the `UploadFile`)

Presence: [0..1]
Definition: The actual file in the `UploadFile` operation.
The file is in Base64 format.
Data Type: Base64Binary

Rule: This element or the EncryptedData element is mandatory in operation `UploadFile`, ignored in other operations. Only one of the Content or EncryptedData elements is allowed.

4.6.1.21 EncryptedData <EncryptedData> (the content in encrypted form)

Presence: [0..1]
Definition: The Content element in encrypted form. The EncryptedData element must decrypt into a valid Content element.
Data Type: EncryptedData (from the XML encryption specification).

Rule: This element or the Content element is mandatory in operation `UploadFile`, ignored in other operations. Only one of the Content or EncryptedData elements is allowed.

4.6.1.22 Signature <Signature> (the digital signature for the ApplicationRequest)

Presence: [0..1]
Definition: Digital signature. This element is created by the *XML Digital Signature* operation by the customer. It is included in this schema as optional element to allow schema validation of the ApplicationRequest element with or without the **Signature**. Its content is specified by the *XML Digital Signature* standard.
Data Type: MaxUnlimitedDSIG (AnyType)
Rule: This element is mandatory when sending any request to the bank as it is used for integrity verification and authentication. This element is defined as optional in the schema because the recipient can remove the signature element after verification of the signature.

4.6.2 ApplicationResponse

This section describes the use of XML element ApplicationResponse field by field.

4.6.2.1 version <version attribute>

Presence: [0..1]

Definition: The version number of the schema according to which this particular XML document has been created. If this attribute is missing, it is assumed the version number is 1. The version number used with schemas documented in this document is 2.

4.6.2.2 CustomerId <CustomerId>

Presence: [1..1]
Definition: Returns the customer identification that was in the corresponding ApplicationRequest.
Data Type: Max16Text

4.6.2.3 Timestamp <Timestamp>

Presence: [1**..**1]
Definition: Time and date when the Application Response Header was created
Data Type: ISODateTime, if no time zone specified, UTC time zone assumed

4.6.2.4 ResponseCode <ResponseCode>

Presence: [1..1]
Definition: The response code given by the bank to indicate the result of the requested operation. The codes are:
Data Type: Code Max16Text

If Code One of the following codes must be used:

Code	Name	Remarks
00	OK.	
01	Pending.	not used
02	SOAP signature error.	signature verification failed
03	SOAP signature error.	certificate not valid for this id
04	SOAP signature error.	certificate not valid
05	Operation unknown.	
06	Operation is restricted.	
07	SenderID not found.	
08	SenderID locked.	
09	Contract locked.	
10	SenderID outdated	
11	Contract outdated	
12	Schemavalidation failed.	
13	CustomerID not found.	
14	CustomerID locked.	
15	CustomerID outdated.	
16	Product contract outdated.	
17	Product contract locked.	
18	Content digital signature not valid.	
19	Content certificate not valid.	
20	Content type not valid.	
21	Deflate error.	
22	Decrypt error.	
23	Content processing error.	
24	Content not found.	
25	Content not allowed.	
26	Technical error.	
27	Cannot be deleted.	
28	[not used]	not used

29	Invalid parameters.	
30	Authentication failed.	
31	Duplicate message rejected.	SOAP.Body.RequestHeader.SenderId + SOAP.Body.ReqhestHeader.RequestId
32	Duplicate ApplicationRequest rejected.	ApplicationRequest.CustomerId + ApplicationRequest.Timestamp
33	TargetID not found	
34	Contract not found	
35	Authorization failed	
36	Technical error, contact bank helpdesk	

4.6.2.5 ResponseText <ResposeText>

Presence: [1..1]
Definition: A text string (human readable) explaining the response code given in the ResponseCode element.
Do not rely on the exact contents of this string, use the ResponseCode value instead.
Data Type: Max80Text

4.6.2.6 ExecutionSerial <ExecutionSerial>

Presence: [0..1]
Definition: The bank returns the ExecutionSerial unique identification code for the operation given by the customer in the ApplicationRequest Header.
Data Type: Max32Text

4.6.2.7 Compressed <Compressed>

Presence: [0..1]
Definition: Compression indicator for the content and compression request for the responses.
Data Type: Boolean
Rule: If this element is present and the content is string *true* (case-sensitive) or 1 it means that the Content is compressed or the requested data should be compressed.
If this element is present and the content is string *false* (case-sensitive) or 0 it means that the Content is NOT compressed or the requested data should NOT be compressed.

Code	Name
True	Payload in Content is compressed
False	Payload in Content is not compressed

4.6.2.8 CompressionMethod <CompressionMethod>

Presence: [0..1]
Definition : Name of the compression algorithm
DataType: Max35Text
Rule: Currently, the only compression algorithm supported is RFC1952 GZIP. Other algorithms may be used but are not covered by this specification. Compression is performed on the original raw data, i.e. before base64 encoding and before placing the data in the Content element.
The table in section 4.6.1.14 lists the codes used and the corresponding algorithms.

4.6.2.9 AmountTotal <AmountTotal>

Presence: [0..1]
Definition: Total sum of amounts in the file.
If the data contained in this request has monetary values, the customer can calculate the total amount of these values and place it in this field. If this element is present, the bank can compare the

values in the data against the value in this element and reject the request if they do not match. The use of this check is to be agreed between the customer and the bank.
This element can also be used in file listings and reports as comment type information to help identify data files. It is easier for the customer to say "file sent last week with total amount around 2.000 euros", instead of "file with FileReference 192830384938". It is up to the bank to decide if it takes the Amount information from this element in the request or if it calculates it from the data file.
Data Type: Double
Rule: There is no requirement for the bank to use this element even if the file type would allow for it.

4.6.2.10 TransactionCount <TransactionCount>

Presence: [0..1]
Definition: The same use as element AmountTotal, but contains the total number of transactions in the data. What "transaction" means varies with type of data, e.g. in C2B pain.001.001.02 payment data TransactionCount is the number of <CdtTrfTxInf> elements.
Data Type: Long
Rule: There is no requirement for the bank to use this element even if the file type would allow for it.

4.6.2.11 CustomerExtension <CustomerExtension>

Presence: [0..1]
Definition: Customer, bank, country or region specific elements not already contained in the schema. This element allows adding new element without changing the ApplicationRequest schema. Both customer and bank must agree on the structure of this element.
Data Type: MaxUnlimitexText

4.6.2.12 FileDescriptors<FileDescriptors> (in response to `DownloadFileList`)

Presence: [0..1]
Definition: An element containing number of **FileDescriptor** elements below.
Data Type: Complex

4.6.2.12.1 FileDescriptor <FileDescriptor>

Presence: [1..n]
Definition: An element containing file attributes below.
Data Type: Complex

4.6.2.12.1.1 FileReference <FileReference>

Presence: [1..1]
Definition: The unique identifier for this file. The identifier is unique in the bank system within one TargetId (folder). File reference is fixed for the entire duration of the file's lifecycle. Therefore, if the client already knows the file references of the desired files (and the file type and the folder) then, for example, it is not mandatory to do a `DowloadFileList` operation each time prior to the `DownLoadFile` operation
Data Type: Max16Text

4.6.2.12.1.2 TargetId <TargetId>

Presence: [1..1]
Definition: The logical folder name where the file(s) of the customer are stored in the bank. A user can have access to several folders.
A customer may want to give their users different views of files and assets that are included in the customer agreement. That can be achieved by organizing file types and assets associated to those file types into separate folders.
Data Type: Max80Text
Rule: Optional for information requests, if omitted the response will cover all files that the user has access to.

4.6.2.12.2 ServiceId <ServiceId>

Presence: [1..1]
Definition: Additional identification information of the Customer, for example a Contract Number, Account Number or similar. This element is used if the CustomerId alone does not give identification that is specific enough to process the request.
Data Type: Max256Text

4.6.2.12.3 ServiceIdOwnerName <ServiceIdOwnerName>

Presence: [0..1]
Definition: Owner of the service identified by ServiceId
Data Type: Max256Text

4.6.2.12.4 UserFileName <UserFileName>

Presence: [0..1]
Definition: A name given to the file by the customer.
The value given in this element in the `UploadFile` operation is stored in the bank and shown in various listings to help the customer identify the file.
Please note that the real identification of a file is the FileReference. The UserFileName field is just comment type information and is not used by bank systems.
Data Type: Max80Text
Rule: This element is mandatory in the operation `UploadFile` and ignored in all other operations.
If missing, request will be rejected, responsecode = “No File Name”

4.6.2.12.5 ParentFileReference <ParentFileReference>

Presence: [0..1]
Definition: A file reference to a file to which this file is related. For example this file could be a status response file to another file. This element indicates the relationship.
Data Type: Max16Text
Rule: This element is mandatory in the operation `UploadFile` and ignored in all other operations.
If missing, request will be rejected, responsecode = “No File Name”

4.6.2.12.6 FileType <FileType>

Presence: [1..1]
Definition: Specifies the type of file in the request. Can also be used as a filter in the operation `DownloadFileList`.
The values accepted in this element are agreed upon between the customer and the bank.
New file types will be added, and they will not affect the schema. A bank specific document will be provided listing commonly used FileTypes.
Data Type: Max40Text
Rule: For ISO messages, the ISO name must be used. This element is mandatory in operation `UploadFile`, optional in `DownloadFileList`, ignored in other operations.

4.6.2.12.7 FileTimestamp <FileTimestamp>

Presence: [1..1]
Definition: The timestamp of the moment the file was created in the bank system.
Data Type: ISODateTime

4.6.2.12.8 Status <Status>

Presence: [1..1]
Definition: The status (state) of the file.
Data Type: Code, Max10Text

If Code One of the following codes must be used:

Code	Name
WFP	Waiting for processing
WFC	Waiting for confirmation
FWD	Forwarded to processing
DLD	Downloaded
DEL	Deleted
NEW	New file
KIN	Key-in

4.6.2.12.9 AmountTotal <AmountTotal>

Presence: [0..1]
Definition: Total sum of amounts in the file.
If the data contained in this request has monetary values, the customer can calculate the total amount of these values and place it in this field. If this element is present, the bank can compare the values in the data against the value in this element and reject the request if they do not match. The use of this check is to be agreed between the customer and the bank.
This element can also be used in file listings and reports as comment type information to help identify data files. It is easier for the customer to say "file sent last week with total amount around 2.000 euros", instead of "file with FileReference 192830384938". It is up to the bank to decide if it takes the Amount information from this element in the request or if it calculates it from the data file.
Data Type: Double
Rule: There is no requirement for the bank to use this element even if the file type would allow for it

4.6.2.12.10 TransactionCount <TransactionCount>

Presence: [0..1]
Definition: The same use as element AmountTotal, but contains the total number of transactions in the data. What "transaction" means varies with type of data, e.g. in C2B pain.001.001.02 payment data TransactionCount is the number of <CdtTrfTxInf> elements.
Data Type: Long
Rule: There is no requirement for the bank to use this element even if the file type would allow for it.

4.6.2.12.11 LastDownloadTimestamp <LastDownloadTimestamp>

Presence: [0..1]
Definition: The timestamp of the moment this file was last downloaded by the customer.
Data Type: ISODateTime
Rule: This element does not exist, the file has not been downloaded.

4.6.2.12.12 ForwardedTimestamp <ForwardedTimestamp>

Presence: [0..1]
Definition: The timestamp of the moment this file was forwarded to processing in the bank.
Data Type: ISODateTime
Rule: This element does not exist, the file has not been forwarded to processing.

4.6.2.12.13 Confirmable <Confirmable>

Presence: [0..1]
Definition: Tells whether the file needs confirmation before being forwarded for processing or allowed to be downloaded.
Data Type: Boolean
Rule: If this element does not exist, it implies the value *false*, i.e. not confirmable.

4.6.2.12.14 Deletable <Deletable>

Presence: [0..1]
Definition: Tells whether the file can be deleted by the customer.
Data Type: Boolean
Rule: If this element does not exist, it implies the value *false*, i.e. not deletable.

4.6.2.12.15 SubStatusCode <SubStatusCode>

Presence: [0..1]
Definition: Some filetypes can have a substatus (substate), eg. Finish Payment Service filetype (LMP300)
Data Type: Max35Text

If Code One of the following codes must be used:

Code	Name
HIGH	High
NORM	Normal

4.6.2.12.16 SubStatusText <SubStatUsText>

Presence: [0..1]
Definition: A text descring the FileSubStatus
Data Type: Max70Text

4.6.2.12.17 MissingTransactions <MissingTransactions>

Presence: [0..1]
Definition: Checksum error indicator for certain filetypes
Data Type: Boolean
Rule: *true* if the validation of the file has discovered that the checksum in the file does not match, otherwise *false*

4.6.2.12.18 SubType <SubType>

Presence: [0..1]
Definition: Valid for some filetypes describibg in more detail what the file content is
Data Type: Max35Text

4.6.2.12.19 FeedbackFileAttributes <FeedbackFileAttributes>

Presence: [0..1]
Definition: Feedback file attributes
Data Type : Complex

4.6.2.12.19.1.1 FeedbackFileReference <FeedbackFileReference>

Presence: [1..1]
Definition: The unique identifier for this file. The identifier is unique in the bank system within one TargetId (folder).
This element is mandatory.
File reference is fixed for the entire duration of the file's lifecycle. Therefore, if the client already knows the file references of the desired files (and the file type and the folder) then it is not mandatory to do a `DowloadFileList` operation each time prior to the `DownLoadFile` operation
Data Type: Max16Text

4.6.2.12.19.1.2 FeedbackFileType < FeedbackFileType>

Presence: [0..1]
Definition: Specifies the file type of the feedback file to be used for download or detailed info. The file types are bank dependent.

Data Type: Max35Text

4.6.2.12.19.1.3 FeedbackFileTypeName <FeedbackTypeName>

Presence: [0..1]
Definition: The name of the feedback fileType.
Data Type: Max80Text

4.6.2.12.19.1.4 FeedbackFileStatus <FeedbackFileStatus>

Presence: [0..1]
Definition: Has the feedback file already been downloaded or not.
Data Type: Code, Max16Text

If Code One of the following codes must be used:

Code	Name
New	the file is new
Downloaded	the file has been downloaded

4.6.2.12.19.1.5 FeedbackFileDate <FeedbackFileDate>

Presence: [0..1]
Definition: The date when the file was created
Data Type: ISODate

4.6.2.12.19.1.6 FeedbackTimestamp <FeedTimestamp>

Presence: [0..1]
Definition: The timestamp of the moment the file was created in the bank system.
Data Type: ISODateTime

4.6.2.12.19.1.7 FeedbackServiceId <FeedbackFileServiceId>

Presence: [0..1]
Definition: Some upload **fileTypes** have a feedback fileType. When a feedback fileType exists the feedback-fields will be provided after a upload has been executed to help the client/user to pinpoint the feedback to the correct upload.
Data Type: Max35Text
Rule: The internal **ServiceId** associated with this feedback file and the feedback information is bank dependent

4.6.2.12.19.2 FileActionHistory <FileActioHistory>

Presence: [0..1]
Definition: A list of actions for the file. Examples of actions are new, download etc.
Data Type: Max16Text

4.6.2.13 UserFileTypes <UserFileTypes> (in response to `GetUserInfo`)

Presence: [1..n]
Definition: The response for `GetUserInfo` contains number of **UserFileType** elements that describe which file types are accessible to this user and their attributes. It is possible to filter the view by specifying**TargetId** and/or **FileType** in the **ApplicationRequest** for `GetUserInfo` request
Data Type: Complex

4.6.2.13.1 TargetId <TargetId>

Presence: [1..1]
Definition: The folder where the file is located.
Data Type: Max80Text

4.6.2.13.2 FileType <FileType>

Presence: [0..1]
Definition: External file type of the file.
Data Type: Max35Text

4.6.2.13.3 FileTypeName <FileTypeName>

Presence: [0..1]
Definition: Display name for the file type
Data Type: Max 80 text
Rule: Empty for files with Upload **Direction**

4.6.2.13.4 Country <Country>

Presence: [0..1]
Definition: Specifies which country where the file type is available in
Data Type: Code, Max3Text, ISO 3166-1 alpha-2 country code

If Code One of the following codes must be used:

Code	Name
FIN	Finland
DNK	Denmark
etc.	etc.

4.6.2.13.5 Direction <Direction>

Presence: [0..1]
Definition: Specifies whether the file is upload or download - type
Data Type: Code, Max16Text

If Code One of the following codes must be used:

Code	Name
Upload	upload direction
Download	download direction

4.6.2.13.6 FileTypeServices <FileTypeServices>

Presence: [0..1]
Definition: Each **UserFileType** element contains number of **FileTypeService** descriptions
Data Type: complex

4.6.2.13.6.1 FileTypeService <FileTypeService>

Presence: [1..n]
Definition: Each `UserfileType` element can contain a number of `FileTypeService` elements that specify which services are available for the file type in the `TargetId` specified by this `UserFileType` element. For example, services associated with an account statement file type are represented by account numbers.
Data Type: complex

4.6.2.13.6.1.1 ServiceId <ServiceId>

Presence: [1..1]
Definition: Internal representation of service identification
Data Type: Max35Text

4.6.2.13.6.1.2 ServiceIdOwnerName <ServiceIdOwnerName>

Presence: [0..1]
Definition: Owner of the service identified by ServiceId
Data Type: Max256Text

4.6.2.13.6.1.3 ServiceIdType <ServiceIdType>

Presence: [0..1]
Definition: (reserved for the future use)
Data Type: Max80Text

4.6.2.13.6.1.4 ServiceIdText <ServiceIdText>

Presence: [0..1]
Definition: Display format for the **ServiceId**
Data Type: Max80Text

4.6.2.14 Content <Content>

Presence: [1..1]
Definition: The actual content, payload in the `DownloadFile` operation.
The file is in Base64 format.
Data Type: MaxUnlimitedBase64

Rule: This element is mandatory in operation `DownloadFile`, optional or ignored in other operations. Only one of the Content and EncryptedData elements is allowed.

4.6.2.15 EncryptedData <EncryptedData> (the content in encrypted form)

Presence: [0..1]
Definition: The Content element in encrypted form. The EncryptedData element must decrypt into a valid Content element.
Data Type: EncryptedData (from the XML encryption specification).

Rule: Only one of the Content and EncryptedData elements is allowed.

4.6.2.16 Signature <Signature> (digital signature of the ApplicationResponse)

Presence: [0..1]
Definition: Digital signature. This element is created by the *XML Digital Signature* operation by the bank. It is included in this schema as optional element to allow schema validation of the ApplicationRequest element with or without the **Signature**. Its content is specified by the *XML Digital Signature* standards.
Data Type: MaxUnlimitedDSIG (AnyType)
Rule: This element is mandatory when sending **ApplicationResponse** to the customer as it is used for integrity verification and authentication of the bank. This element is defined as optional in the schema because the recipient can remove the signature element after verification of the signature.

4.7 Sample Messages and Application Requests

4.7.1 Request SOAP Message

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
<env:Header>
      <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd" env:mustUnderstand="1">
            <wsse:BinarySecurityToken wsu:Id="bst_Zkt4E6PpC4aTK272"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
1.0.xsd" ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-
profile-1.0#X509v3" EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
soap-message-security-1.0#Base64Binary">MIIDF...(most Base64 removed for
clarity)...8Xx60=</wsse:BinarySecurityToken>
            <dsig:Signature xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
                  <dsig:SignedInfo>
                        <dsig:CanonicalizationMethod
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
                        <dsig:SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha256"/>
                        <dsig:Reference URI="#Body_SJqHqDhuSvW7UkFo">
                              <dsig:Transforms>
                                    <dsig:Transform Algorithm="http://www.w3.org/2001/10/xml-
exc-c14n#">
                                          <exc14n:InclusiveNamespaces
xmlns:exc14n="http://www.w3.org/2001/10/xml-exc-c14n#" PrefixList=""/>
                                    </dsig:Transform>
                              </dsig:Transforms>
                              <dsig:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha256"/>

<dsig:DigestValue>4yzYxO6f0W9wu4YkQf4zayxTiLs=</dsig:DigestValue>
                        </dsig:Reference>
                  </dsig:SignedInfo>

<dsig:SignatureValue>PBVGxh7x2kzFYnkrL15zMqtLa5RHuqvRVEFcIbQzaivGnjJJTE3fOozbAb3st1ZHTjwCykX
/ZWP+NPNe9KvtaB959Jve3zUZbnrA1Deyg7GNAQQaDfbnGxW6uooyQOp+xwOsoIqDBVp83nigdKfsEhOKt6EWmMug+Ov
w6V8Cxvk=</dsig:SignatureValue>
                  <dsig:KeyInfo>
                        <wsse:SecurityTokenReference xmlns:wsse="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
1.0.xsd" wsu:Id="str_Pqf55eQFnl5t1jOT">
                              <wsse:Reference URI="#bst_Zkt4E6PpC4aTK272"
ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-
1.0#X509v3"/>
                        </wsse:SecurityTokenReference>
                  </dsig:KeyInfo>
            </dsig:Signature>
            <wsu:Timestamp xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-utility-1.0.xsd">
                  <wsu:Created>2008-04-14T10:07:31Z</wsu:Created>
                  <wsu:Expires>2008-04-14T10:08:31Z</wsu:Expires>
            </wsu:Timestamp>
      </wsse:Security>
</env:Header>
<env:Body wsu:Id="Body_SJqHqDhuSvW7UkFo" xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
      <cor:uploadFilein xmlns:cor="http://bxd.fi/CorporateFileService">
            <java:RequestHeader xmlns:java="java:fi.bxd.model">
                  <java:SenderId>2457785447</java:SenderId>
                  <java:RequestId>1234567</java:RequestId>
                  <java:Timestamp>2008-04-14T13:07:26.371+00:00</java:Timestamp>
                  <java:Language>FI</java:Language>
                  <java:UserAgent>TestClient 1.00</java:UserAgent>
                  <java:ReceiverId>BANKCODE</java:ReceiverId>
            </java:RequestHeader>
            <java:ApplicationRequest xmlns:java="java:fi.bxd.model">PD94b... (most Base64
removed for clarity)...lc3Q+</java:ApplicationRequest>
      </cor:uploadFilein>
```

```
</env:Body>
</env:Envelope>
```

4.7.2 Response SOAP Message

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<env:Header>
      <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd" env:mustUnderstand="1">
            <wsse:BinarySecurityToken wsu:Id="bst_q81bYMu9uCbcF6oq"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
1.0.xsd" ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-
profile-1.0#X509v3" EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
soap-message-security-1.0#Base64Binary">MIIB6... (most Base64 removed for
clarity)...CmFP5</wsse:BinarySecurityToken>
            <dsig:Signature xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
                  <dsig:SignedInfo>
                        <dsig:CanonicalizationMethod
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
                        <dsig:SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha256"/>
                        <dsig:Reference URI="#Body_y5x1WczgM0hNrIJb">
                              <dsig:Transforms>
                                    <dsig:Transform Algorithm="http://www.w3.org/2001/10/xml-
exc-c14n#">
                                          <exc14n:InclusiveNamespaces
xmlns:exc14n="http://www.w3.org/2001/10/xml-exc-c14n#" PrefixList=""/>
                                    </dsig:Transform>
                              </dsig:Transforms>
                              <dsig:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha256"/>

<dsig:DigestValue>Q6hTqYuZuVJunJEPb3pYkh9qeCU=</dsig:DigestValue>
                        </dsig:Reference>
                  </dsig:SignedInfo>

<dsig:SignatureValue>jcXxJbI71MF70V1f/Gaxb97bU9G3RXBXpp/OFANhwlC54mh0KGIXm/p3myWZH0bpylNE0yh
bq80bsfHMXkZfSw==</dsig:SignatureValue>
                  <dsig:KeyInfo>
                        <wsse:SecurityTokenReference xmlns:wsse="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
1.0.xsd" wsu:Id="str_mHKU7V4pBVDxWXxN">
                              <wsse:Reference URI="#bst_q81bYMu9uCbcF6oq"
ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-
1.0#X509v3"/>
                        </wsse:SecurityTokenReference>
                  </dsig:KeyInfo>
            </dsig:Signature>
            <wsu:Timestamp xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-utility-1.0.xsd">
                  <wsu:Created>2008-04-11T09:58:19Z</wsu:Created>
                  <wsu:Expires>2008-04-11T09:59:19Z</wsu:Expires>
            </wsu:Timestamp>
      </wsse:Security>
</env:Header>
<env:Body wsu:Id="Body_y5x1WczgM0hNrIJb" xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
      <cor:downloadFileListout xmlns:cor="http://bxd.fi/CorporateFileService">
            <java:ResponseHeader xmlns:java="java:fi.bxd.model">
                  <java:SenderId>2457785447</java:SenderId>
                  <java:RequestId>1234567</java:RequestId>
                  <java:Timestamp>2008-04-14T13:07:31.121+00:00</java:Timestamp>
                  <java:ResponseCode>00</java:ResponseCode>
                  <java:ResponseText>OK.</java:ResponseText>
                  <java:ReceiverId>BANKCODE</java:ReceiverId>
            </java:ResponseHeader>
            <java:ApplicationResponse xmlns:java="java:fi.bxd.model">PD94b... (most Base64
removed for clarity)...lPg==</java:ApplicationResponse>
      </cor:downloadFileListout>
```

```
</env:Body>
</env:Envelope>
```

4.7.3 UploadFile ApplicationRequest

```
<?xml version="1.0" encoding="UTF-8"?>
<ApplicationRequest version="2" xmlns="http://bxd.fi/xmldata/">
<CustomerId>4956859506</CustomerId>
<Timestamp>2008-04-14T13:07:27.843+03:00</Timestamp>
<Environment>TEST</Environment>
<TargetId>target</TargetId>
<Compression>true</Compression>
<CompressionMethod>GZIP</CompressionMethod>
<SoftwareId>TestSoft 1.00</SoftwareId>
<FileType>pain.001.001.02</FileType>
<Content>eJydV... (most Base64 removed for clarity)...Zoy0=</Content>
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
      <SignedInfo>
            <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-
20010315#WithComments"/>
            <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha256"/>
            <Reference URI="#xpointer(/)">
                  <Transforms>
                        <Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-
signature"/>
                  </Transforms>
                  <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha256"/>
                  <DigestValue>5PsKL0uX0HMiXN4704l28qdK0NQ=</DigestValue>
            </Reference>
      </SignedInfo>

<SignatureValue>NOHlZ5FZVZuprzKtDzViVfi/lIRjlftImW7wfmf8HVm8ojxdfralzmLIRVK1I0d6RV1HEx09Qy/k
JhBdAKnJnKYeKBMuDfEDRtZu+9jI4Ijwm4Q0i2CNeA8Qh8ijq++kkGDl7X5oF+fTvVQs+IFV1gZWdgTmT8yfasVskxyO
G3Q=</SignatureValue>
      <KeyInfo>
            <X509Data>
                  <X509Certificate>MIIDF... (most Base64 removed for
clarity)...97CPc=</X509Certificate>

            </X509Data>
      </KeyInfo>
</Signature>
</ApplicationRequest>
```