Luminor

Luminor Web Service specification

Version 1.0.2

This document defines how a file (e.g. a payment file) which will be sent to the bank is digitally signed by the content owner. It also describes how a file received from the bank is digitally signed by Luminor. Digital signature is used for authentication and integrity control of the file. This model is used in Luminor File Transfer solution.

Contents

Version	Date	Description of changes
Version 1.0	May 20, 2020	Document published
Version 1.0.1	June 16, 2020	Endpoints update
Version 1.0.2	September 23, 2021	Added the list of returned errors

1. Introduction

Luminor Web Service is used for exchanging files and messages via a network. Luminor Web Service is for corporate customers is called shortly WS. It will be the entry point for Baltic customers and will support various file transfer protocols and different file formats, including XML SEPA payments in the ISO20022 standard.

This specification describes how to protect file content using a digitally signed Secure Envelope, which is transported over different available communication protocols. All files exchanged through WS are digitally signed according to the specification in this document independent of the channel/protocol used.

This specification of securing the file content will not change or alter the security in any of the communication channels.

2. Background

Files which are to be exchanged with a bank are in a specified file format and layout according to the requested service (e.g. payments). Normally the file format specification does not include security elements. To enable end-to-end security of a file, Luminor introduces a content signature model called Secure Envelope which is described in this document. With the Secure Envelope, security will be part of the content, i.e. related to the content owner instead of the communication channel owner.

The new security model is based on PKI and XML digital signature technologies, which both are global standards. The XML structure, called Secure Envelope, is described in following chapters. Any file, whether it is XML, ASCII or binary, can be transported by using the Secure Envelope. Once the customer has added his/her digital signature to the Secure Envelope and thereby to the file, the file can be securely transported via any channel/protocol to Luminor.

3. Requirements

The sending of digitally signed files requires:

- Software which creates the Secure Envelope according to this specification
- Software to create a digital signature for the Secure Envelope
- File content (i.e. the payment file)
- PKI keys, i.e. the certificate. The certificate can be obtained via certificate web service call provided by from Luminor.

4. The Secure Envelope structure and elements used

The Secure Envelope described here is an XML structure following a public schema. The Secure envelope contains the file to be sent and is digitally signed. The schema is based on an open specification and used in other contexts as well. The Secure Envelope is part of an open specification for Web Services, publicly available via Bankers' Association in Finland.

The schema contains several elements, some mandatory and some optional. Luminor requires information for some optional elements, like Command and Signature.

The rule is that one Secure Envelope is present for each physical file (the payload). In case files are compressed, these files still form one physical file of the same file type. The customer signature authorisation is then connected to that file, type of file and service.

4.1 ApplicationRequest

When sending a file to the bank, the Secure Envelope uses a schema called ApplicationRequest.

It will use following namespaces from:

https://www.w3.org/2001/XMLSchema
https://www.w3.org/TR/2002/REC-xmldsig-core-20020212/xmldsig-core-schema.xsd

Some of elements in the schema are not used by Luminor FTC and thus not shown.

The elements used when uploading a file to a bank are marked with (M) if they are mandatory for Luminor, even if they are not a mandatory in the schema. Elements used should not be left empty.

ApplicationRequest Schema and elements used is shown in **Appendix D.**

4.2 ApplicationResponse

When receiving a file from the bank the Secure Envelope uses a schema called ApplicationResponse.

ApplicationResponse schema is always used when files are transported from Bank to Customer. It enables XML digital signature to any file type.

The payload file is inserted into the element called Content. The payload file is base64 coded to make it independent of the ApplicationResponse schema. It has some supporting elements for information purposes, like returned CustomerID, Timestamp, ResponseCode, Compression etc. Application response is used to enable file integrity and to validate that the message is received from the correct party.

ApplicationResponse Schema is shown in **Appendix E**.

5. Security

5.1 XML Digital signature

XML Digital Signature must be added to the Secure Envelope for all files sent to or received from Luminor FTC.

5.2 Type of digital signature

XMLDsig is of type Enveloped, i.e. it signs the whole XML structure (ApplicationRequest or ApplicationResponse). It is specified by W3C.

The signature must contain the following elements with associated child elements: SignedInfo, SignatureValue and KeyInfo. KeyInfo should contain X509Data as well as the child element X509Certificate, including the customer's public key of the signing certificate.

See more information from:

https://www.w3.org/TR/xmldsig-core/
https://en.wikipedia.org/wiki/XML_Signature

5.3 Download of Signer ID certificate

The Signer ID certificate must be downloaded by using web service request to certificate service.

Certificate service endpoint and WSDL are defined in **Appendix C**

6 The file transfer process

6.1 Sending a file to bank

The file transfer process can be divided in different parts. The three main ones are:

1. Creating the file content to be sent e.g. pain.001 XML payment
2. Locking the file, i.e. signing the file content with XML digital signature, using the content signing security key
3. Moving the signed file to the communication software, which connects to a bank server and transports the signed file into Luminor using a file communication protocol security solution and key(s).

Depending on the infrastructure and legacy solutions in the company, the steps above can be carried out in different systems or using just one solution.

The first step: The file content is usually created in a corporate legacy and the file type must follow the specifications of the requested service (i.e. a message implementation guideline).

Step nr 2 is described below:

After creation of the payload file, a Secure Envelope following the ApplicationRequest schema is created, according to the instructions in Appendix D.

The payload will be entered in the element called ‘Content’ of type base64Binary.

All needed elements must have specified values according to Appendix D. After that the Secure Envelope with payload can be locked, i.e. digitally signed by using customer content signing PKI key. Once it is locked the payload is protected against any changes from the moment of signing, until it is received by Luminor FTC File Transfer. This is called integrity control.

The signature also uniquely defines the signing party of the payload, i.e. the content owner. Authentication of the customer is based on the content signature.

The signed Secure Envelope can be now sent to Luminor by supported file transfer protocol.

Banking software is used to initiate a secure communication, using customer specific communication security key(s) received from Luminor. The content signing key is always a PKI key.

See Figure 4 below, which describes the steps in creating a payload file and a Secure Envelope, signing the Secure Envelope and sending it by any communication protocol to Luminor.

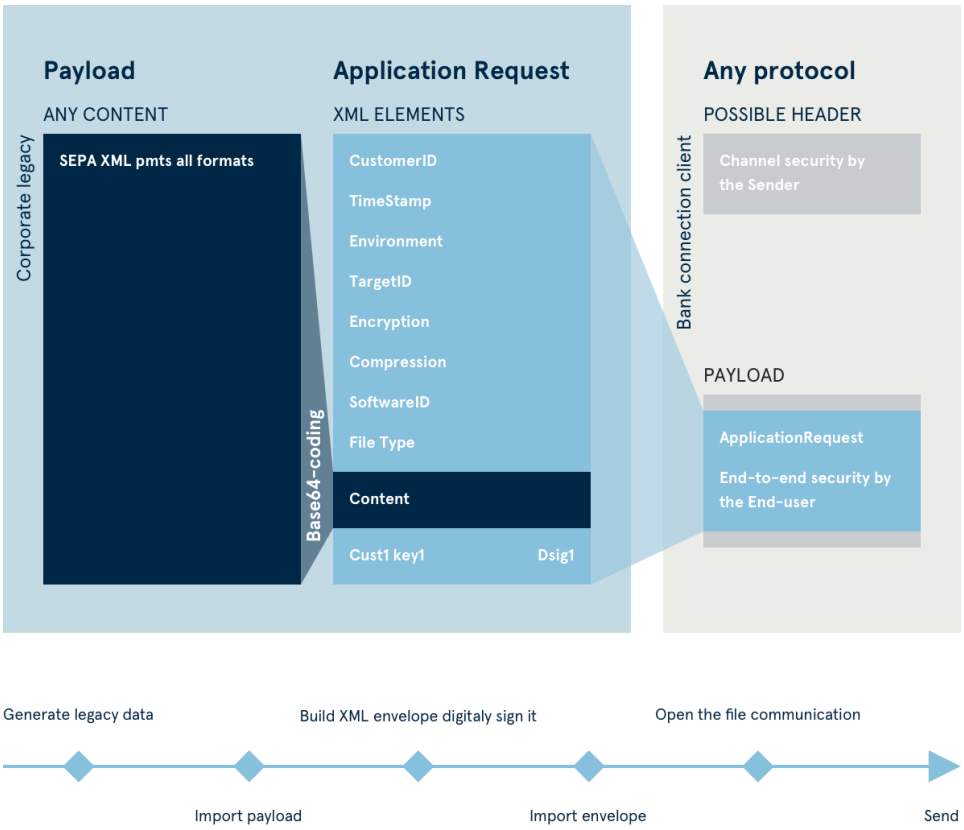


Figure1

Normally a legacy system at the bank which received the file from the customer will send the customer a status message. A status message file may indicate that a sent file has been received by the bank, state the result after validation of the payload, etc. There may be multiple status messages created during the legacy process. See chapter 6.2.

6.2 Receiving a file from bank

Files from the bank are made available for customer to retrieve them (push-pull protocol).

The receiving of a file from Luminor is also based on a Secure Envelope, following ApplicationResponse schema. The signer in this case is Luminor and the receiver has a possibility to authenticate the real content signer by verifying the signature validity. Luminor provides the CA keys needed for this verification.

The signature ensures that the payload content created by Luminor is not changed by anyone. Any, even the smallest change of the content, would invalidate the signature. If so, happens there is an additional channel/protocol independent integrity and authentication control mechanism for the customer. However, the ApplicationResponse schema is different from the ApplicationRequest schema (see Appendix E).

The receiver of the Secure Envelope should first verify the signature, and then look into the elements in the Secure Envelope. If the response contains a (requested) payload, it may be compressed following the values in the respective elements.

The response file may also contain an error message, in which case the ResponseCode element in the Secure Envelope contains a value other than zero. The different error messages based on values are described in Appendix A.

The actual payload file can be extracted from the Secure Envelope by carrying out base64 decoding to the file in the Content element

Luminor Web service provides following services:

- uploadFile - upload single file
- downloadFileList - get a list of files to download, file reference collection
- downloadFile - download a single file by file reference
- deleteFile
- getUserInfo - get file types which are accessible to the user

Each service implements ApplicationRequest call and provides ApplicationResponse result described by XSDs.

7. Service providers

A 3rd party acting on behalf of the End-Customer

A company or an entrepreneur may outsource the handling of cash management payables and receivables to a so-called third party. The 3rd party is thus acting on behalf of the endcustomer. An example of this is a book-keeping agency which manages several customers' payables/receivables.

Another example could be a company with a "payment factory", where one legal unit manages payables and receivables on behalf of other units in the same company.

If a 3rd party is to create payment files using the end-customer's debit account on behalf of the end-customer, the end-customer must give the 3rd party power of attorney (PoA). Then the end-customer adds the Signer ID of the 3rd party to the service agreement. The PoA must be registered and be available at Luminor.

When the 3rd party sends a payment file, the signature in the Secure Envelope belongs to 3rd party, but the debit account in payment file may belong to end-customer. The authorization to use the debit account is verified before execution of the payment transactions, and a valid PoA must exist. If the PoA will be cancelled, it must be registered in Luminor, respectively.

When a 3rd party is sending files to Luminor, they must have an agreement with Luminor for file transfer. By that they will receive their own Sender ID and one or several Sign ID's/Certificates. If they will use any of the services, like payment services for their own purposes, they will need an agreement for that service as well.

8.Vocabulary and abbreviations used

CustomerID	Identifier provided by Luminor ("Sender ID") to identify a sender/customer in a file communication agreement
Signature	A digital signature signing the content using a PKI key. An authorised signer may use the signature to pre-confirm payments. Also, signing the SOAP envelope in Web Services.
Signer	A bank customer who has an agreement and authorisation to create a digital signature for the payload (e.g. a payment file), by using a customer-specific certificate/PKI key. In the response file Luminor is the signer.

Service ID	Identifier provided by Luminor to identify a file content.
File **(Content, Payload, Message)**	The message, which is to be exchanged, e.g. a Credit Transfer initiation message. It should follow the Message Implementation Guideline (MIG) published by Luminor. The file content is sometimes referred to as the payload, i.e. the meaningful data to be transported.
File type	A pre-defined string entered in the Secure Envelope, introducing the type of file in the Content element, to be sent or retrieved to/from the specific service in the bank.
3rd party	A bank customer who has a power of attorney from another bank customer, to act on the latter's behalf. An example is a book-keeping agency making payments on behalf of another customer.
ApplicationRequest	A name of a schema used for the Secure Envelope containing the file and signature to be sent to the bank.
ApplicationResponse	A name of a schema used for the Secure Envelope containing the file and signature to be received from the bank.
PKI	Public Key Infrastructure. A standard process for managing asymmetric key handling in digital signatures.

Appendix A Web Service Error codes

Error Code	Name	Remarks
00	OK	
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	
07	SenderID not found	
08	SenderID locked	
09	Contract locked	
10	Sender ID outdated	
11	Contract outdated	
12	Schema validation failed	XML Envelope is not valid
13	Customer ID not found	
14	Customer ID locked.	
15	Customer ID 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	Decompression not possible
22	Decrypt error	Decryption not possible
23	Content processing error	
24	Content not found	
25	Content not allowed	
26	Technical error	
27	Cannot be deleted	
29	Invalid parameters	Invalid value found in Envelope
30	Authentication failed	Sender ID or Signer ID unknown
31	Duplicate message rejected	
32	Duplicate application request rejected	

Appendix B File types for Web Service

All file types should be supported by the receiving software and communication system.

Upload	**File type**	**Options**
Payment file	PAIN001	
Download	**File type**	**Options**
Feedback file	PAIN002	
Account Notification	CAMT054	A Status NEW SubType CAMT054 B Status NEW SubType CAMT054_C C Status NEW SubType CAMT054_D
Account statement	CAMT053	
Account report	CAMT052	A Status NEW SubType ALL B Status NEW SubType NEW C Status NEW SubType BALANCE
SWIFT MT940	SW940	
SWIFT MT941	SW941	
SWIFT MT942	SW942	

Appendix C Web Service Endpoints

Production

Soap Requests endpoints:
https://ftc.luminoropenbanking.com/v1/ft-services/CertificateService
https://ftc.luminoropenbanking.com/v1/ft-services/CorporateFileService

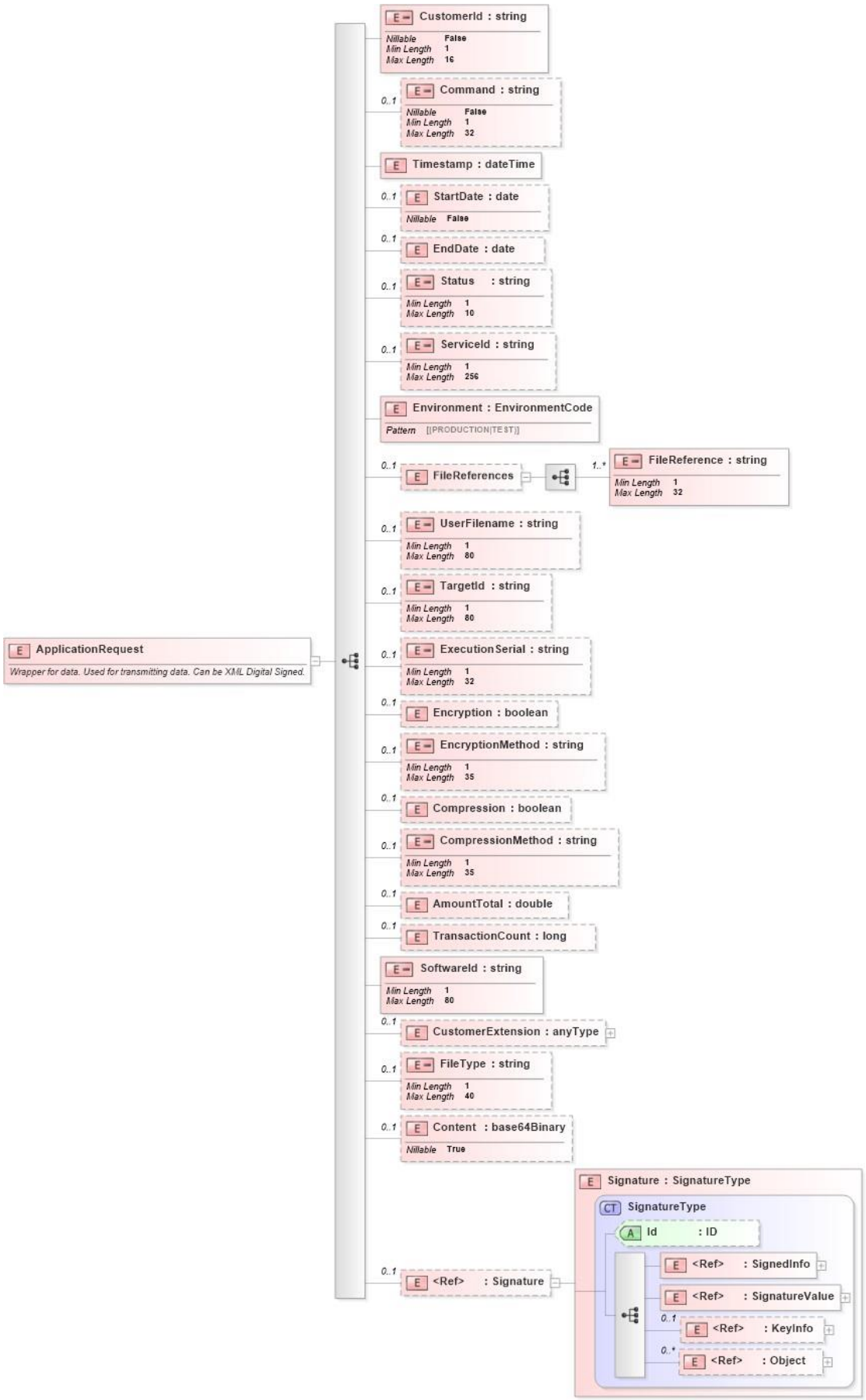
WSDLs:
https://ftc.luminoropenbanking.com/v1/ft-services/CertificateService.wsdl
https://ftc.luminoropenbanking.com/v1/ft-services/CorporateFileService.wsdl

Sandbox

Sandbox Soap Requests endpoints:
https://filetransfer.developer.luminoropenbanking.com/v1/ft-services/CertificateService
https://filetransfer.developer.luminoropenbanking.com/v1/ft-services/CorporateFileService

Sandbox WSDLs:
https://filetransfer.developer.luminoropenbanking.com/v1/ft-services/CertificateService.wsdl
https://filetransfer.developer.luminoropenbanking.com/v1/ft-services/CorporateFileService.wsdl

Appendix D ApplicationRequest



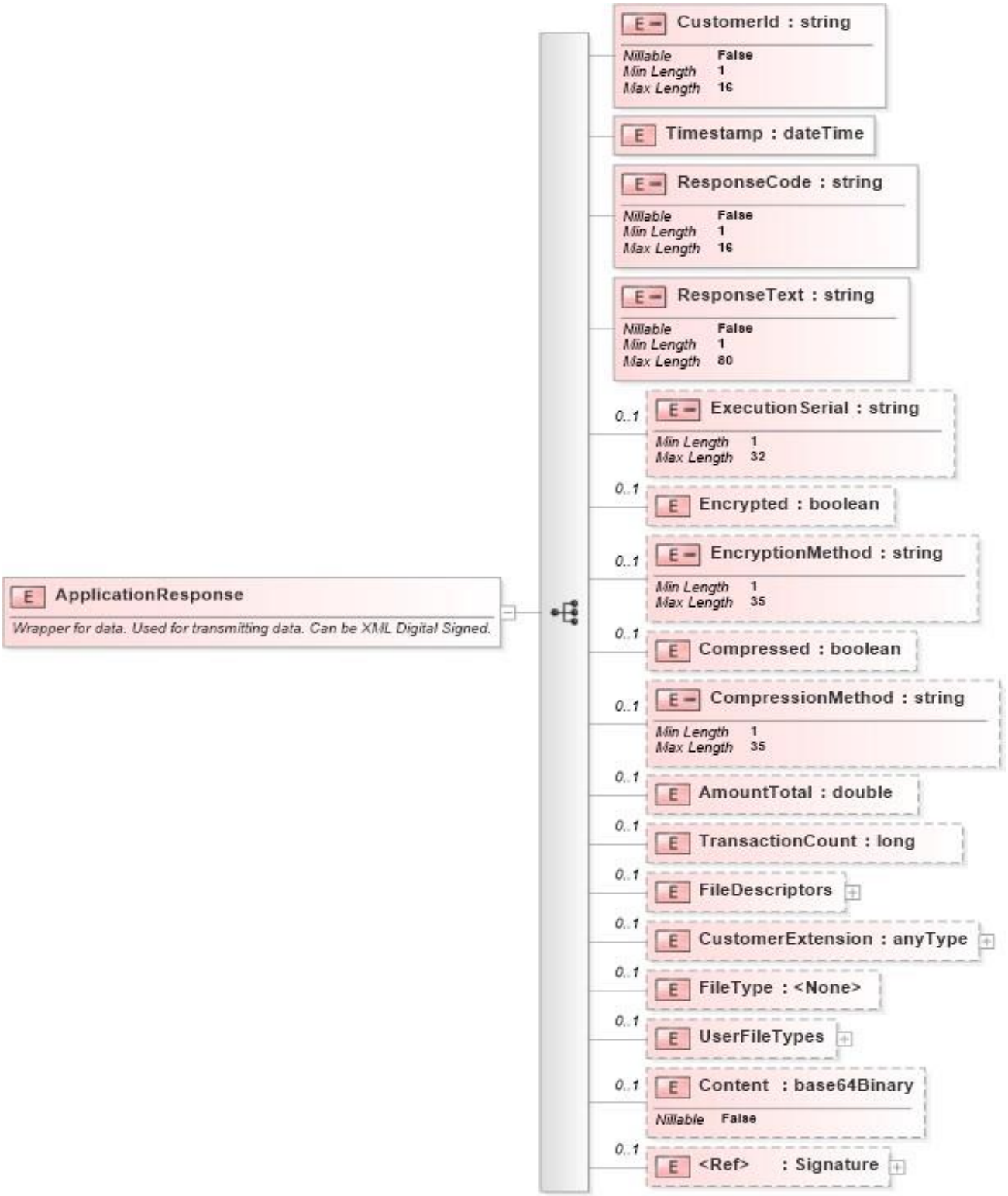
Elements in ApplicationServiceRequest

Element	M/O	Description
CustomerId	M	The content for this element is found in the file communication customer agreement as “Sender ID”. Each agreement contains only one Sender ID. In some special cases, when the signer of the content uses another party for transporting the signed file, this field contains the Sender ID of the other party.
Command	M	The command is dependent on the case used. • UploadFile, when sending a file to the bank • DownloadFile, when getting file from • DownloadFileList, to obtain downloadable content list by criteria set by either StartDate, EndDate, Status, ServiceId • getUserInfo, when get information on file types which are accessible to the user
Timestamp	M	Creation time of the ApplicationRequest Secure Envelope. UTC or local time. Valid values Current date -7 days, +1 day
StartDate	O	Date Filtering criteria in DownloadFileList Operation
EndDate	O	Date Filtering criteria in DownloadFileList operation
Status	M	Content Filtering criteria in DownloadFileList operation One of the following codes are possible to use: NEW, DLD, ALL
ServiceId	O	Filtering criteria in DownloadFileList operation, for those file types that has Service ID in use
Environment	M	Must be “PRODUCTION”
FileReferences		Mandatory in DownloadFile request Unique identification of the file which is the target of operation Obtained with downloadlist operation
TargetId	M	The content for this element is found in the File Communication Customer Agreement as “Signer ID”. Each agreement may contain several Signer ID’s. Each Signer ID is connected to a certain content signing certificate. The Signer ID authenticates the identity and the authorization to the content or file type according to the agreement. This element is mandatory in all operations. There can be only one Signer ID in a SecureEnvelope

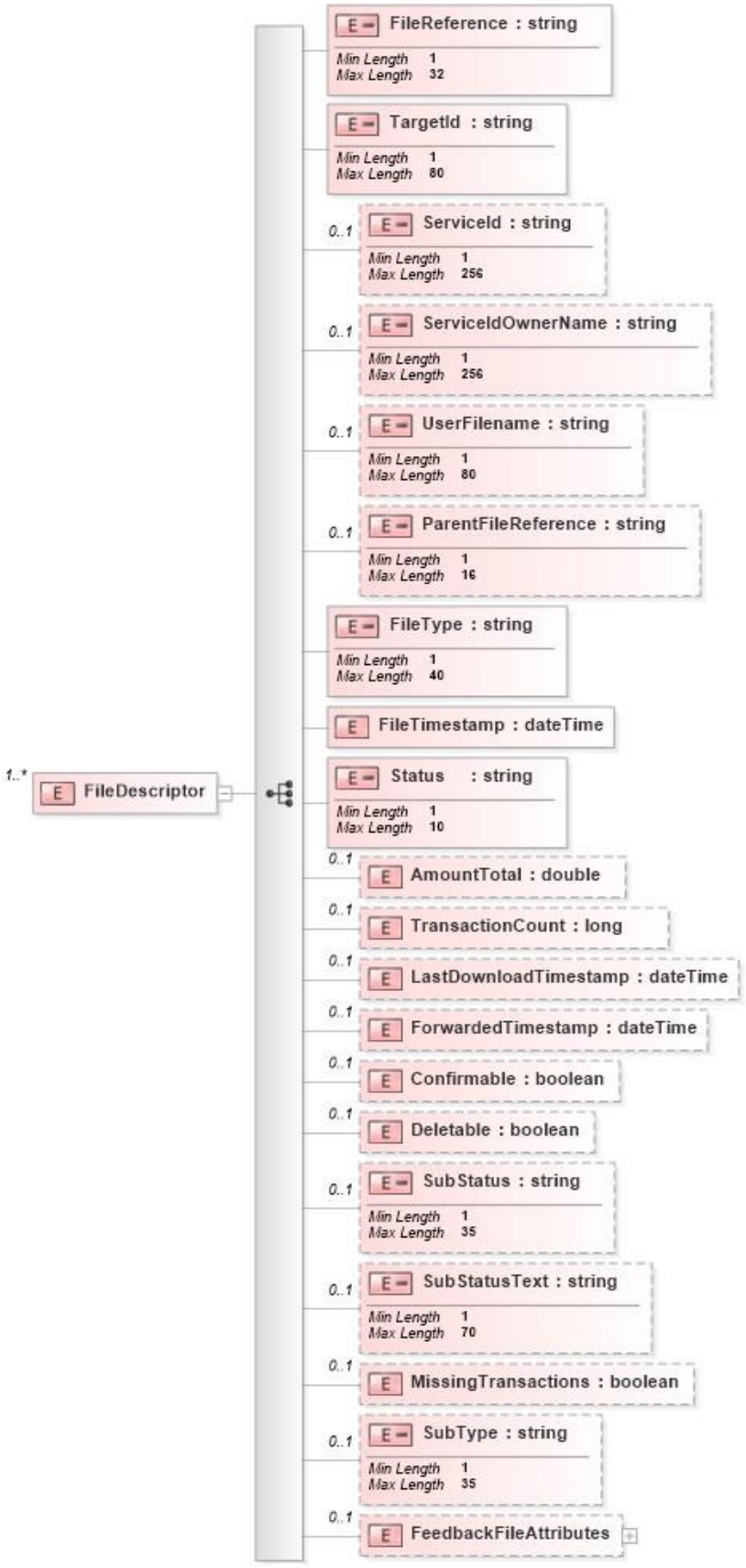
ExecutionSerial	O	The content for this element is found in the File Communication Customer Agreement as “Signer ID”. Each agreement may contain several Signer ID’s. Each Signer ID is connected to a certain content signing certificate.
		The Signer ID authenticates the identity and the authorization to the content or file type according to the agreement. This element is mandatory in all operations. There can be only one Signer ID in a SecureEnvelope ‘\’
Encryption	O	“false”
EncryptionMethod	O	Not used
Compression	O	“false” or “true” The value indicates if the uploaded content is compressed or not If set to true, then the compression method element must also have a value. Files are to be compressed before possible encryption and before base64 coding. It is recommended to compress files bigger than 1 MB. Files from bank that are bigger than 1 MB will always be compressed. Files which original size is bigger than 10 MB should sent always in compressed form In the DownloadFile operation the field can be used for request compression
CompressionMethod	O	If compression is set to “true” the value in CompressionMethod is “GZIP”. It means that the Compression algorithm is RFC1952 GZIP
SoftwareId	M	Contains the name and version of the client-side software which generated the ApplicationRequest Secure Envelope.
FileType	M	Specifies uniquely the type of file in the request. Available file types are specified in Appendix B
Content	M	The actual file in the UploadFile operation. The content is in Base64 format. Specifies uniquely the type of file in the request. Available file types are specified in Appendix B

Signature	M	This element is created by the signature operation by the customer. Signature content is specified by the XML Digital Signature standard. 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, before schema

Appendix E ApplicationResponse



File Descriptor



Elements in ApplicationServiceResponse

Element	M/O	Description
CustomerId	M	The content for this element is found in the file communication customer agreement as “Sender ID”. Each agreement contains only one Sender ID. In some special cases, when the signer of the content uses another party for transporting the signed file, this field contains the Sender ID of the other party.
Timestamp	M	Creation time of the ApplicationRequest Secure Envelope. UTC or local time. Valid values Current date -7 days, +1 day
ResponseCode	M	The content of this field is 0, if, no error appeared, otherwise the specific error code is presented describing the error as defined in Appendix A.
ResponseText	O	„OK” if no error , otherwise the error description according to Appendix A.
ExecutionSerial	O	An identifier given by the customer is returned in response. Otherwise 0 (zero) One of the following codes are possible to use: NEW, DLD, ALL
Encryption	O	“false”
EncryptionMethod	O	Not used
Compression	O	“false” or “true” The value indicates if the uploaded content is compressed or not If set to true, then the compression method element must also have a value. Files are to be compressed before possible encryption and before base64 coding. It is recommended to compress files bigger than 1 MB. Files from bank that are bigger than 1 MB will always be compressed. In the DownloadFile operation the field can be used for request compression
CompressionMethod	O	If compression is set to “true” the value in CompressionMethod is “GZIP”. It means that the Compression algorithm is RFC1952 GZIP

FileDescriptors	M	Collection of FileDescriptors See single FileDescriptor element in appendix D Will be present in response for DownloadFileList
FileType	M	Specifies uniquely the type of file in the request. Available file types are specified in Appendix B
Content	M	The actual file content of the operation. The content is in Base64 format
		Specifies uniquely the type of file in the request. Available file types are specified in Appendix B
Signature	M	This element is created by the signature operation by the customer. Signature content is specified by the XML Digital Signature standard. 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, before schema

Elements in Elements in FileDescriptor

Element	M/O	Description
FileReference	M	Example: 00001500922006030316603001006827
TargetId	M	Example: 0012345678
FileType	M	Example Payments feedback: PAIN002 List of file types available can be found from Appendix B
FileTimestamp	M	Example: 2006-03-03T00:00:00.0Z
Status	M	Example: NEW or DLD

List of returned errors

Error code	Error string	Message
4	SOAP_SIGNATURE_ERROR_CERTIFICATE_NOT_VALID	Signature certificate not valid
40001	SOAP_SIGNATURE_ERROR_CERTIFICATE_FAILED	Signature validation failed
5	OPERATION_UNKNOWN	Account not found = <account>
6	OPERATION_IS_RESTRICTED	Operation is not available for supplied credentials
6001	OPERATION_IS_DISABLED	File deletion disabled (Customer endpoint)
6001	OPERATION_IS_DISABLED	File deletion disabled (FTC endpoint)
7	SENDER_ID_NOT_FOUND	Sender ids should be integer numbers, got: <senderId>
13	CUSTOMER_ID_NOT_FOUND	Customer ids should be integer numbers, got: <customerId>
18	CONTENT_DIGITAL_SIGNATURE_NOT_VALID	Signature validation failed
18	CONTENT_DIGITAL_SIGNATURE_NOT_VALID	Digital signature validation failed
19	CONTENT_CERTIFICATE_NOT_VALID	Signature certificate not valid
19001	CONTENT_CERTIFICATE_DUPLICATED	Unexpected error, attempted to revoke multiple certificates, serial serialNumber
20	CONTENT_TYPE_NOT_VALID	Cannot process file type <filetype>
230001	CONTENT_PROCESSING_ACCOUNT_ERROR	Account Services Reference not present! <AcctSvcrRef> is missing
230002	CONTENT_PROCESSING_UNZIP_ERROR	Cannot unzip gzip file
230003	CONTENT_PROCESSING_BAFII_ERROR	Serializing BranchAndFinancialInstitutionIdentification4 failed
230004	CONTENT_PROCESSING_CB_ERROR	Serializing CashBalance3 failed
230005	CONTENT_PROCESSING_OI_ERROR	Serializing OriginalGroupInformation20 failed
230006	CONTENT_PROCESSING_OPI_ERROR	Serializing OriginalPaymentInformation1 failed
230007	CONTENT_PROCESSING_PI_ERROR	Serializing PartyIdentification32 failed
230008	CONTENT_PROCESSING_PF_ERROR	Serializing PropertyFile failed
230009	CONTENT_PROCESSING_RE_ERROR	Serializing Report2Entry failed
230010	CONTENT_PROCESSING_JAXB_ERROR	{many possible messages}
230011	CONTENT_PROCESSING_GZIP_METHOD_ERROR	Unsupported compression method - <compressionMethod>
230012	CONTENT_PROCESSING_SAX_CONFIG_ERROR	Unable to SAX parse
230013	CONTENT_PROCESSING_SAX_ERROR	SAX parse error
230014	CONTENT_PROCESSING_ASIC_INVALID_ERROR	asic-e container invalid
230015	CONTENT_PROCESSING_ASIC_SUBJECT_ERROR	asic-e signature subject is invalid
230016	CONTENT_PROCESSING_ASIC_SIGN_COUNT_ERROR	<signaturesSize> signatures in asic-e container
230017	CONTENT_PROCESSING_ASIC_FILE_COUNT_ERROR	<dataFilesSize> files in asic-e container
230018	CONTENT_PROCESSING_REP_ID_ERROR	Report ID is missing
230019	CONTENT_PROCESSING_REP_ACC_ERROR	Report Account is missing
230020	CONTENT_PROCESSING_REP_ACC_ID_ERROR	Report Account ID is missing
230021	CONTENT_PROCESSING_PKCS_ERROR	unexpected pkcs10 subject: + subject
230026	CONTENT_PROCESSING_SECRET_VALUE_ERROR	Error processing secret value

230027	CONTENT_PROCESSING_BIC_ERROR	Document is missing mandatory BIC field
230028	CONTENT_PROCESSING_CCY_ERROR	Document is missing mandatory Ccy attribute in <InstdAmt> tag
230029	CONTENT_PROCESSING_XML_PARSE_ERROR	Could not parse application request xml
26	TECHNICAL_ERROR	Handling request failed
260001	TECHNICAL_ARCHIVE_ERROR	Archived UploadFileLog not found by ID = <id>
260002	TECHNICAL_MISSING_FILE_ERROR	File not found by messageId = <messageId>
260003	TECHNICAL_MT_FILE_ERROR	Failed to parse MT file
260004	TECHNICAL_CERT_ERROR	Failed to serialize application response
260005	TECHNICAL_CERT_SER_ERROR	Failed to serialize cert application response
260006	TECHNICAL_JAXB_SER_ERROR	Failed to serialize jaxbElement
260007	TECHNICAL_MAP_OTJ_ERROR	Failed to write object into json
260008	TECHNICAL_MAP_JTO_ERROR	Failed to write json into object
260009	TECHNICAL_MAP_OTP_ERROR	Failed to write object into json
260010	TECHNICAL_CERT_FORMAT_ERROR	Formatting X509Certificate failed
260011	TECHNICAL_MAP_TO_LIST_ERROR	Mapping ResultSet to List failed
260012	TECHNICAL_MAP_TO_OPTIONAL_ERROR	Mapping ResultSet to Optional failed
260013	TECHNICAL_IS_TRUE_ERROR	<value> is not true
260014	TECHNICAL_NOT_NULL_ERROR	<value> is null
260015	TECHNICAL_FILE_ERROR	Unknown file
260016	TECHNICAL_CERT_REVOKE_ERROR	Valid certificate to revoke not found
260017	TECHNICAL_JSON_PARSE_ERROR	Message with error in JSON
260018	TECHNICAL_CUST_EXIST_ERROR	Customer with customerId = <t24CustomerId> already exist
260019	TECHNICAL_ORGANIZATION_ERROR	Invalid organization data: <organization>
260020	TECHNICAL_NO_ACCOUNT_ERROR	Customer with customerId = <customerId> has no accounts
260021	TECHNICAL_ACCOUNT_ERROR	Customer with customerId = <customerId> has invalid account data <account>
260022	TECHNICAL_FILE_TYPE_ERROR	Unknown file type (<fileTypeCode>)
260023	TECHNICAL_BIC_ERROR	Cannot transform source bic <bic> to correct BIC for customer ID <customerId> and country <country>
260023	TECHNICAL_NO_FILE_ERROR	File not found by message id = <messageId>, fileType = <filetype>
260026	TECHNICAL_CERT_USER_ERROR	Soap message and application request signature certificates belong to different users
260027	TECHNICAL_AGREEMENT_STATUS_ERROR	Wrong status
260028	TECHNICAL_AGREEMENT_NUMBER_ERROR	Agreement not found by agreement number = <agreementNumber>
260029	TECHNICAL_AGREEMENT_ID_ERROR	Agreement not found by agreement id = <agreementId>
260030	TECHNICAL_BALANCE_ERROR	Failed to parse balance
260031	TECHNICAL_PAYMENT_ERROR	Failed to parse original payment information
260032	TECHNICAL_GROUP_INFO_ERROR	Failed to parse original group information
260033	TECHNICAL_SERVICER_ERROR	Failed to parse servicer
260034	TECHNICAL_DOC52_PARSE_ERROR	Failed to parse document
260035	TECHNICAL_DOC53_PARSE_ERROR	Failed to parse document
260036	TECHNICAL_OWNER_PARSE_ERROR	Failed to parse owner
260037	TECHNICAL_REPORT_ENTRY_ERROR	Failed to parse report entry
260038	TECHNICAL_DATETIME_PARSE_ERROR	Failed to parse xml dateTime: <dateTime>
260039	TECHNICAL_REPORT_TYPE_ERROR	Unknown report type - <type>
260040	TECHNICAL_UNMARSHAL_SCHEMA_ERROR	Unmarshal create schema error
260041	TECHNICAL_UNKNOWN_STATUS_ERROR	Unknown status - <currentStatus>

260042	TECHNICAL_KEY_ALIAS_ERROR	key by alias not found
260043	TECHNICAL_FORMAT_CERT_ERROR	Certificate formatting failed
260044	TECHNICAL_PARSE_PKCS_ERROR	request user doesn't match cert user. Tech userId=<technicalUserId>
260045	TECHNICAL_CTRLSUM_ERROR	CtrlSum and transaction total sums can not be different
260046	TECHNICAL_NBOFTXS_ERROR	NbOfTxs and transaction count can not be different
260047	TECHNICAL_FILE_DIRECTION_ERROR	Unknown file type (<fileTypeCode>) and direction (<direction>) combination
260048	TECHNICAL_REQUEST_LOG_ERROR	RequestLog not found by requestId = <requestId>
27	CANNOT_BE_DELETED	You cannot remove main Customer (id = <customerId>) for Agreement (id = <agreementId>)
29	INVALID_PARAMETERS	A problem has occurred with reading pkcs#10 csr
290001	INVALID_NO_COUNTRY_PARAMETERS	Country is not specified in the subject
290002	INVALID_NO_CUSTOMER_NAME_PARAMETERS	Customer name is not specified in the subject
290003	INVALID_NO_USER_ID_PARAMETERS	User id is not specified in the subject
290004	INVALID_COUNTRY_PARAMETERS	PKCS10 csr subject country is <subject>, but the agreement is for <country>
290005	INVALID_CUSTOMER_NAME_PARAMETERS	PKCS10 csr subject customer name is <customerName>, but the agreement is for <customerMainName>
290006	INVALID_USER_ID_PARAMETERS	User id is <userId>, but the SERIALNUMBER in PKCS10 csr subject is <userId>
290007	INVALID__ENV_PARAMETERS	Environment is <environment>, but the message is for <environment>
290008	INVALID_CERT_PARAMETERS	No valid certificates found
290009	INVALID_CERT_COUNT_PARAMETERS	Multiple valid certificates found, specify a serialNumber of the certificate you want to revoke
290010	INVALID_MESSAGE_ID_PARAMETERS	MessageId is not unique
290011	INVALID_FOLDER_PARAMETERS	Unknown folder - <folder> for key <key>
290012	INVALID_CERT_CONTENT_PARAMETERS	Certificate content is required
290013	INVALID_ASIC_USER_PARAMETERS	Technical user not found
290014	INVALID_CERT_SERIAL_PARAMETERS	Serial number must be an integer
290015	INVALID_REQUEST_TARGET_ID_PARAMETERS	Target id is mandatory
290016	INVALID_REQUEST_ACCESS_ACC_PARAMETERS	User <userId> does not have access to <accountNumber> account, <fileType> fileType
290017	INVALID_REQUEST_STATUS_PARAMETERS	Not valid status = <status>
290018	INVALID_REQUEST_START_DATE_PARAMETERS	Start date <startDate> can't be in the future
290019	INVALID_REQUEST_END_DATE_PARAMETERS	End date <endDate> can't be before start date <startDate>
290020	INVALID_REQUEST_FILE_REF_PARAMETERS	One file reference is allowed
290021	INVALID_REQUEST_ACCESS_TARGET_PARAMETERS	User <id> doesn't have access to <targetId> targetId, user's targetId is <userTargetId>
30	AUTHENTICATION_FAILED	signature certificates' user doesn't match with application request's customerId.
33	FILE_GENERATION_FAILED	Building CSV failed with error
34	S3_BUCKET_ERROR	Missing bucket <configuredBucket>
35	SIGNATURE_ERROR	Failed to read signature
350001	SIGNATURE_INACTIVE_ERROR	Certificate is not active. id: id
350002	SIGNATURE_INVALID_ERROR	Certificate is invalid. id: id
350003	SIGNATURE_INVALID_DATE_ERROR	Found certificate (id: <id>) is not valid. <validFrom> - <now> - < validTo>
350004	SIGNATURE_RSA_ERROR	signing method algorithm uri is not RSA: algorithm

350005	SIGNATURE_INVALID_MAIN_ERROR	Certificate validation failed
350006	SIGNATURE_NOT_FOUND_ERROR	Certificate with fingerprint not found: <fingerprint>
350007	SIGNATURE_NOT_MATCH_ERROR	Certificate in db doesn't match it's fingerprint. id: <id>
350008	SIGNATURE_RSA_KEY_ERROR	certificate's key algorithm is not RSA: <algorithm>
350009	SIGNATURE_SIGN_TAG_ERROR	Could not find <Signature> XML tag
350010	SIGNATURE_SIGN_COUNT_ERROR	Expected one Signature element, got length
350011	SIGNATURE_REF_XML_ERROR	Could not find <Reference> XML tag
350012	SIGNATURE_REF_COUNT_ERROR	Expected one Reference in Signature, got length
350014	SIGNATURE_CORE_INVALID_ERROR	Signature is invalid
350015	SIGNATURE_USER_ERROR	Active user not found by id = <userId>
350016	SIGNATURE_SOAP_COUNT_ERROR	expected one certificate in a soap request, got <length>
350019	SIGNATURE_VALIDATE_ERROR	<Various messages about certificate signature error>
350020	SIGNATURE_VALIDITY_ERROR	<Various messages about certificate signature error>