EUROPEAN COMMISSION
DIRECTORATE-GENERAL
INFORMATICS
Information systems Directorate

European Commission

Interface Control Document
Open e-TrustEx version 2.4.0

Date:	20/12/2018
Version:	1.000
Authors:	Sonia Tafaro Alice Vasilescu D'Orazio Sandro Anamaria BATRINU (DIGIT-EXT)
Revised by:	Cristian CHIRIAC (DIGIT-EXT)
Approved by:	Ilyasse SETTI (DIGIT-EXT)
Public:	
Reference Number:	

TABLE OF CONTENTS

TABLE OF FIGURES

LIST OF TABLES

Document History

Version	Date	Comment	Modified Pages
1.000	05/12/2018	ICD Open e-TrustEx 2.4.0.	

1. INTRODUCTION

1.1. Background

The e-TrustEx action was launched by DIGIT in 2010 to support public administrations in the implementation of EU policies by offering them a platform for secure information exchange. At the time this document is written, e-TrustEx is already being used in several business domains:

- In the procurement domain, it enables the European Commission and Member States to exchange procurement documents in digital format with their Suppliers.
- In the competition domain, it enables the organisations involved in competition cases to exchange documents with the European Commission;
- In the legislative domain, it enables the parliaments in the EU to exchange legislative documents with the European Commission.
- In the Justice domain, it enables citizens through the use of the Justice portal to submit cross border claims.
- In the Health domain, it enables tobacco companies to submit product ingredients to the European Commission.

In all cases, e-TrustEx plays the role of interoperable mediator between the back-offices of exchange parties.

The e-TrustEx platform is already available on Joinup (European Commission, 2011) in Open Source so that it can be freely reused by Public Administrations. Its main elements, coarse grained view, are shown in the model below.

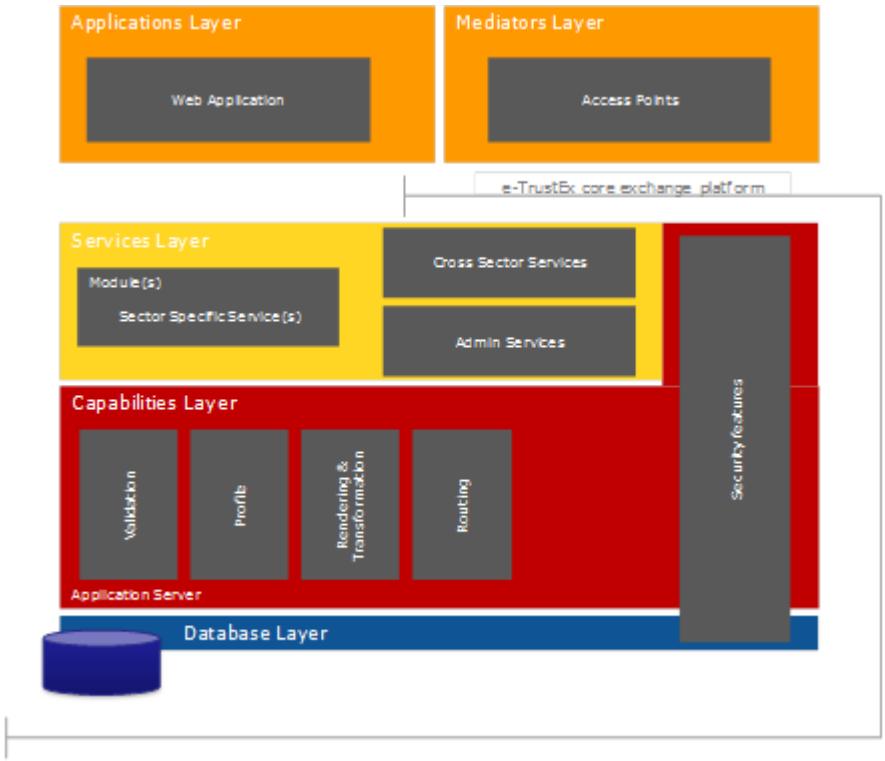


Figure 1 - e-TrustEx platform

The e-TrustEx core exchange platform is divided into the following interface-mediated layers:

- **Services Layer:** Set of services of e-TrustEx which can be directly accessed, via well-defined interfaces, by the application and mediators layers. Information exchanged across the service layer is wrapped within structured or semi-structured message structures. The services included in this layer are categorised as follows:
    - **Cross Sector Services:** Highly reusable, cross-cutting services which can be used in several sectors and do not change very regularly over time e.g. inbox service, retrieve document service, etc.;
    - **Sector Specific Services:** These services are created and directly associated with a specific business process of a given business sector e.g. submit invoice for the eProcurement business domain. They therefore change more regularly and have less reuse potential across sectors than the cross sector services. Nonetheless, these services are usually highly reusable within their specific sector.
    - **Administration Services:** These services are used by administrators and central business owners to administer and configure e-TrustEx.
- **Capabilities Layer:** Services often perform similar functions such as validation, archiving, etc. As opposed to implementing these capabilities in the services themselves, they are implemented in this layer so that they can be reused. These capabilities can be thought as internal services that are not visible to the application and mediators layers.

1.2. Purpose of the Interface Control Document

The purpose of this document is to detail the interfaces provided by e-TrustEx platform - Trusted Exchange Platform.

1.3. Scope

The scope of this document covers the documentation of e-TrustEx interfaces.

It describes the different possible interactions, the services involved in these interactions and the messages exchanged.

Related references to the technical endpoint definitions (WSDL), use cases and document structures (XSD) are also provided in this document for each interaction.

1.4. What is a service

For the purpose of the ICD, a service is defined as an abstract resource that performs specific actions (a.k.a. business-level operations, or simply, operations) in the scope of one or several business processes. Within the context of e-TrustEx, the most natural and familiar way for a service to operate is via the exchange of documents between the entity providing the service and the entity requesting its use (and vice-versa). Therefore, each business process can be decomposed in a set of services made up of one or more atomic operations. The operands of each operation are documents which are exchanged and processed by these entities. Each Operation will either finish in a 'success' or 'error' state[1].

[1] For the sake of transparency and reliability, e-TrustEx enables the users to follow-up the state of each operation using the StatusRequest service. For more information on the states of a request the reader should consult Status Request section.

The ICD details the "what" and the "how" about the services exposed by the e-TrustEx platform. The "what" details what services are available and what should be expected from each service from a functional and non-functional viewpoint, while the "how" details how these services are implemented from a technical viewpoint and how they should be used to yield the expected result.

1.5. Audience

The ICD is intended for a diverse audience and is aimed at everyone interested in understanding the interface of e-TrustEx from a functional or a technical viewpoint. The target audience for this document includes business domain and technical experts.

According to RUP (see[REF9]), the roles listed hereunder are part of the target audience:

- Implementers of the services of e-TrustEx, for an understanding of the interface provided by the service and the behaviour its clients should expect.
- Implementers of the consumers of these services, for an understanding of the technical interface exposed by e-TrustEx, the inputs required by each service and also their outputs. Additional information is also available about the evolution of these services.
- Designers of services, in understanding the relationship between specifications and the relationship between services and the specifications they implement.
- Testers of the services, to understand the functionality and quality aspects of the service model.

1.6. Definitions, Acronyms and Abbreviations

Term	Description
e-TrustEx	Name of the Trusted Exchange Platform currently being developed by the European Commission, DG-Informatics.
HTTP (Hyper Text Transfer Protocol)	A TCP-based application-layer protocol used for communication between Web servers and Web clients.
HTTPS	Secure version of the HTTP protocol. A different default port and an additional encryption/authentication layer between HTTP and TCP are used.
Receiver	A Party which is referred to as the recipient of a message sent through e-TrustEx.
Sender	A Party which is referred to as the origin of a message sent through e-TrustEx.
SOA (Service Oriented Architecture)	An architectural style where existing or new functionalities are accessible by means of services, without knowing the underlying technology.
SOAP (Simple Object Access Protocol	A lightweight XML-based messaging protocol used to encode the information in Web service request and response messages before sending them over a network.
SSL (Secure Sockets	A protocol for transmitting private information via the Internet by means of a cryptographic system.

Layer)	
UBL	Universal Business Language (UBL) is a library of standard electronic XML business documents such as purchase orders and Invoices. UBL was developed by an OASIS Technical Committee with participation from a variety of industry data standards organizations. UBL is designed to plug directly into existing business, legal, auditing, and records management practices. It is designed to eliminate the re-keying of data in existing fax- and paper-based business correspondence and provide an entry point into electronic commerce for small and medium-sized businesses. UBL version 2.0 was approved as an OASIS Committee Specification in October 2006 and has been publicly released.
WS (Web Service)	A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-process able format (specifically WSDL).
WSDL	WSDL (Web Service Description Language) is an XML-based service description on how to interface using a web service.
XML	XML (Extensible Mark-up Language) is the standard messaging format for business communication, allowing companies to connect their business systems with those of customers and partners using the existing Internet infrastructure.
XSD	XML schema definition language describes the structure of an XML document.
SAD	Software Architecture Document
TLS	Transport Layer Security

1.7. References

#	Document	Contents outline
[REF1]	e-TrustEx Use Cases	**Page containing the use case diagram** corresponding to the services offered by e-TrustEx.
[REF2]	Fault Events	**e-TrustEx SOAP faults** description.
[REF3]	e-TrustEx WSDLs	**Page containing all WSDLs** corresponding to the e-TrustEx services.
[REF4]	e-TrustEx XSDs	**Directory containing the XML schemas** for the e-TrustEx messages.
[REF5]	e-TrustEx software architecture document	**e-TrustEx** SAD
[REF6]	OASIS Universal Business Language v2.0	**Universal Business Language v2.0**
[REF7]	XML Schema 1.1	**XML Schema**
[REF8]	ISO Schematron	**ISO/IEC 19757 – 3 : 2006**
[REF9]	Rational Unified Process version 7.5	**RUP roles** that are part of the target audience.
[REF10]	EC_CodeList_e-TrustEx.xls	Code list tables
[REF11]	ETSI TS 101 903 V1.3.2	**ETSI Standard** for XML Advanced Electronic Signature (XAdES)
[REF12]	CEN BII	**CEN Business Interoperability Interfaces** for public procurement in Europe
[REF13]	ebXML	**Electronic Business** using **eXtensible Markup Language**
[REF14]	SOAP 1.1	**Simple Object Access Protocol 1.1**
[REF15]	GLN	**Global Location Number (**also known as EAN)
[REF16]	NAL	**Named Authority Lists**
[REF17]	SOAP Messages with Attachments	
[REF18]	MTOM	**Message Transmission Optimization Mechanism**
[REF19]	WSDL 1.1	**Web Services Description Language**
[REF20]	XML 1.0	**EXtensible Markup Language**
[REF21]	WS-I Basic Profile Version 1.1	
[REF22]	HTTP Over TLS	
[REF23]	Hypertext Transfer Protocol 1.1	
[REF24]	HTTP Authentication: Basic and Digest Access Authentication	

[REF25]	Open e-TrustEx	**Joinup link for Open e-TrustEx**

2. ARCHITECTURAL PROBLEM STATEMENT

The key architectural problem which e-TrustEx aims at solving is the interoperability between the broad collection of systems used by the Public Administrations at European, national and regional level. The business purpose is the secure exchange of digital structured and unstructured documents from system to system via standardised interfaces.

It allows Public Administrations to replace paper documents or files stored on DVDs and CDs by system-to-system exchange of information, using a technologically advanced platform.

The figure below provides the answer to the above problem statement. This figure depicts the high-level conceptual architecture of e-TrustEx. Note that the difference between the External Interaction Layer and the Internal Interaction Layer is one of conceptual nature and is provided in order to better understand the business processes. Technically both layers are identical.

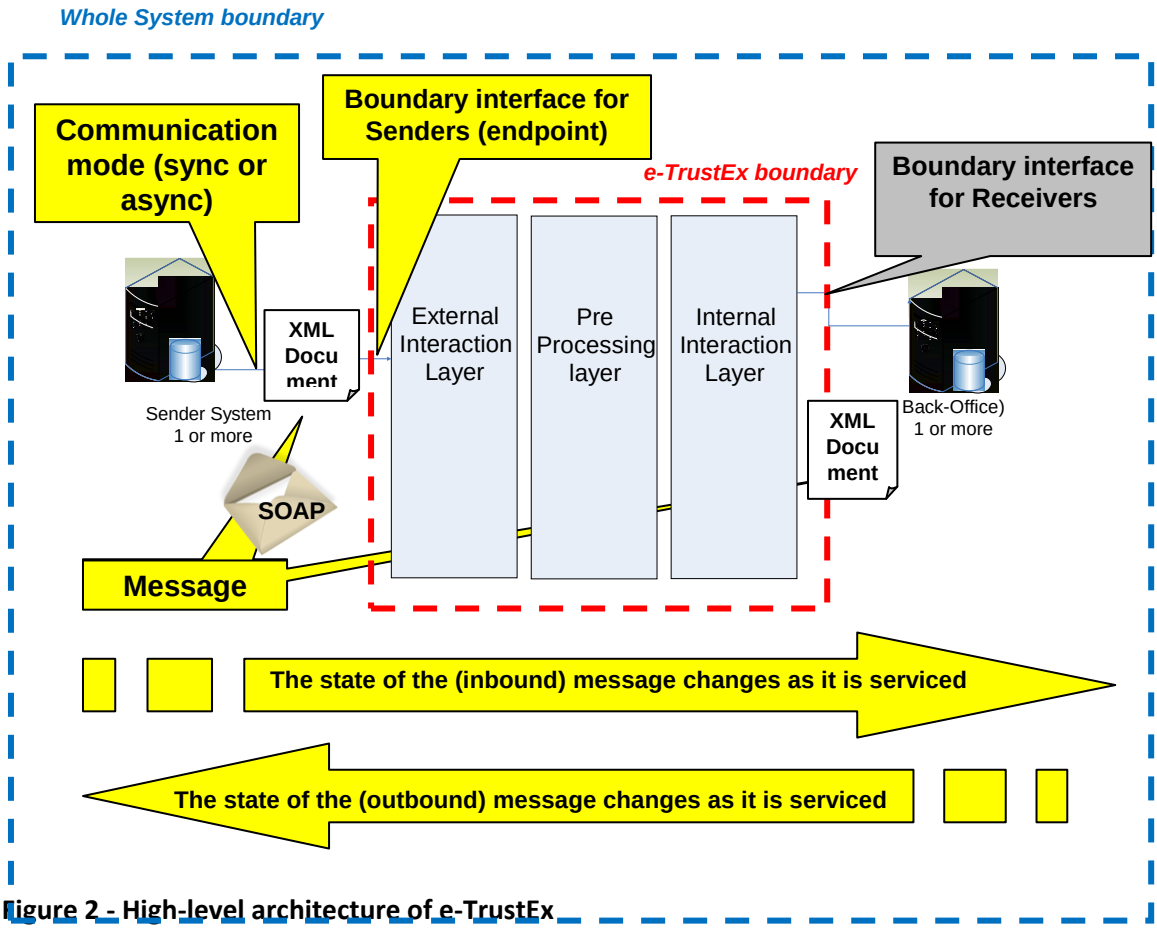


Figure 2 - High-level architecture of e-TrustEx

The e-TrustEx platform is composed by a set of cooperating layers, each of which deals with the tasks described below:

- The External Interaction Layer implements the endpoints that interact with the Senders' Information Systems.
- The Pre-Processing Layer, implements the application of a number of business rules on the requests and on the documents to be exchanged as well as the routing to the correct destination.
- Internal Interaction Layer, implements the endpoints that interact with the back-offices of the Public Administrations.

Each layer uses the functions implemented by the other. This separation of responsibilities between layers provides the flexibility and robustness required by a system positioned right in-between the back-offices of the Public Administrations and the Information Systems of the Senders. This is the visual illustration of the most important

architectural concept of e-TrustEx. As a mediator system handling requests from both sides, e-TrustEx is the enabler of information exchange in a loosely coupled fashion.

The fundamental operand of the system is the set of documents exchanged between Senders and Receivers in the form of messages. Below is the conceptual description of the processing of a document submitted by a Sender to a Receiver through e-TrustEx from start to finish:

1) The Sender starts by creating the electronic document using its Information System.
2) Since the document is already in electronic format, the tasks to be performed by the Sender are the conversion to the appropriate format (i.e. as specified by the interface of e-TrustEx) and its transmission to the Receiver via the e-TrustEx specific end-point.
3) The Information System of the Sender communicates with e-TrustEx over a secured network channel via a well-defined interface.
4) Every document crossing e-TrustEx's boundary has a specific message type. An acknowledgment of reception is immediately sent back to the Sender.
5) Once successfully received, the message is processed by e-TrustEx and then forwarded to the back-office of the Receiver (more specifically the responsible delegate system that manages the Document Workflow).
6) The result of this processing yields the business value to both entities (e.g. Attached Document in the state processed).
7) Once the document processing is complete, its status is updated based on an Application Response sent by the Receiver and the Application Response is made available to the Sender. Both Inbox Request Service and Query Request Service enable the Sender to consult the list of documents that were posted by the Receiver. The contents of these documents can then be retrieved using the Retrieve Request service.

To increase transparency during the aforementioned multi-step processing workflow, the Sender will be able to submit Read Requests to e-TrustEx (e.g. Status Requests). This enables the Sender to track the actual state of the request processing and, once available, fetch the contents of the business response following an Inbox Request and Retrieve Request.

The example detailed above focuses on the submission of a document by the Sender to a Receiver. However, e-TrustEx also enables Receivers to submit documents to Senders (e.g. Application Responses).

3. ARCHITECTURAL DESIGN

3.1. Data model

3.1.1. XML encoding

The content of the XML documents (except the message ID) may contain non-ASCII characters, like Norwegian æ ø å, or French ê è é. To avoid encoding errors, the Sender system should respect the UTF-8 encoding. The encoding declaration should follow this example:

<?xml version="1.0" encoding="UTF-8" ?>

3.1.2. XML Schema Definitions

A well-defined library of reusable data components is used throughout several documents enabling an enterprise-wide semantic model instead of a per-document silo definition. The data model adopted by e-TrustEx is based on UBL (see [REF6]) and is synchronised with the CEN WS/BII specifications (CEN Workshop on Business Interoperability Interfaces for public procurement in Europe, see [REF12]).

The UBL Technical Committee uses the W3C XSD (see [REF7]) standard to specify the UBL business documents. The ebXML CCTS (see [REF13]) is the foundation of their design. In addition to standard XML business documents, UBL provides formal Naming and Design Rules and Customisation Guidelines. All elements and types are defined in a specific namespace depending on their "nature": Basic Information Entity, Association Information Entity or Aggregate Information Entity. Main documents simply reuse these elements and attributes. The XSD ref mechanism is used to point to previously declared elements in their namespace.

The default UBL namespaces are used in most documents exchanged in e-TrustEx's Asynchronous Services. However in order to simplify the data model, these documents have been customised into an e-TrustEx specific data model. This syntax and semantic model must be respected in every document submitted to e-TrustEx. Therefore, the XML instances to be sent to e-TrustEx are derived from the UBL Standard but are not the UBL standard documents. When a UBL namespace is used, the customised instance must be validated against the customised schema.

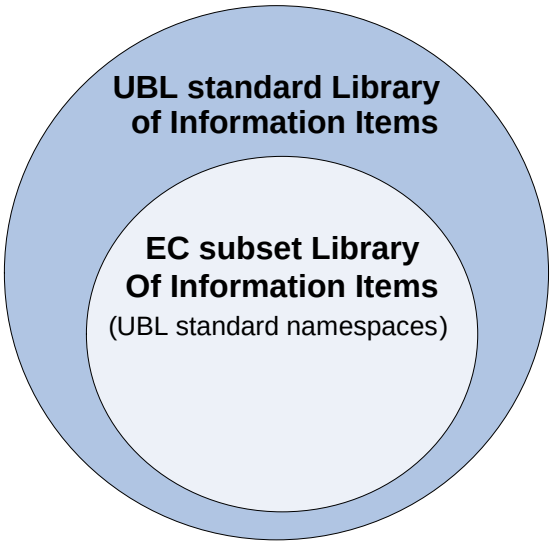


Figure 3 - EC UBL subset: The default UBL document namespaces are used in most documents exchanged using e-TrustEx Asynchronous Services ›nal Services.

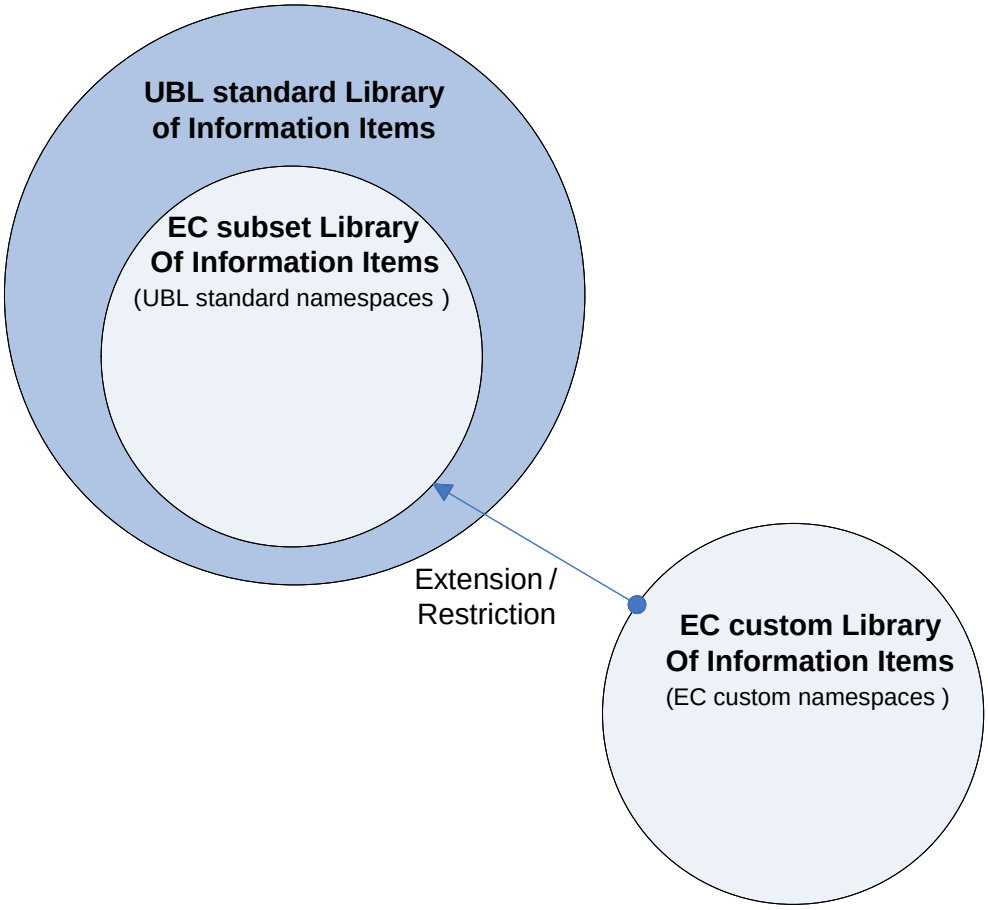


Figure 4 - EC UBL subset: Custom document namespaces are used in most documents exchanged using e-TrustEx Read / Operational Services

In both cases, the backbone of the documents exchanged via e-TrustEx is a Common Library of information elements. This Library contains general descriptions of all information elements that exist within the different business documents. The philosophy behind the Library is to achieve the highest reusability for the different information elements. This enables the reuse of the same constructs in several contexts within the same or different business documents.

The diagram below shows the dependencies among the several schema modules of most UBL XML document schemas (see [REF7]).

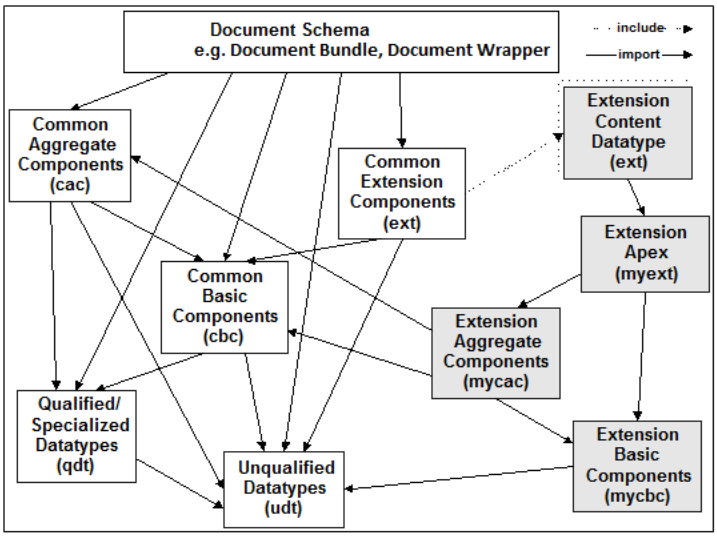


Figure 5 - UBL Documents Architecture

3.1.3. Message validation

E-TrustEx implements a multi-phase document validation for maximum flexibility in configuring and updating business rules (such as compliance to predefined code-lists).

- In the first validation phase, the document is checked against its XSD (structural and lexical validation).
- In the second validation phase (value validation), documents are checked against specific business rules mostly implemented using a Schematron XSLT (see [REF8]). This second phase has no impact on the schemas.
- In the third validation phase (semantic interpretation), documents are checked against specific business rules implemented in the code of the e-TrustEx platform itself.

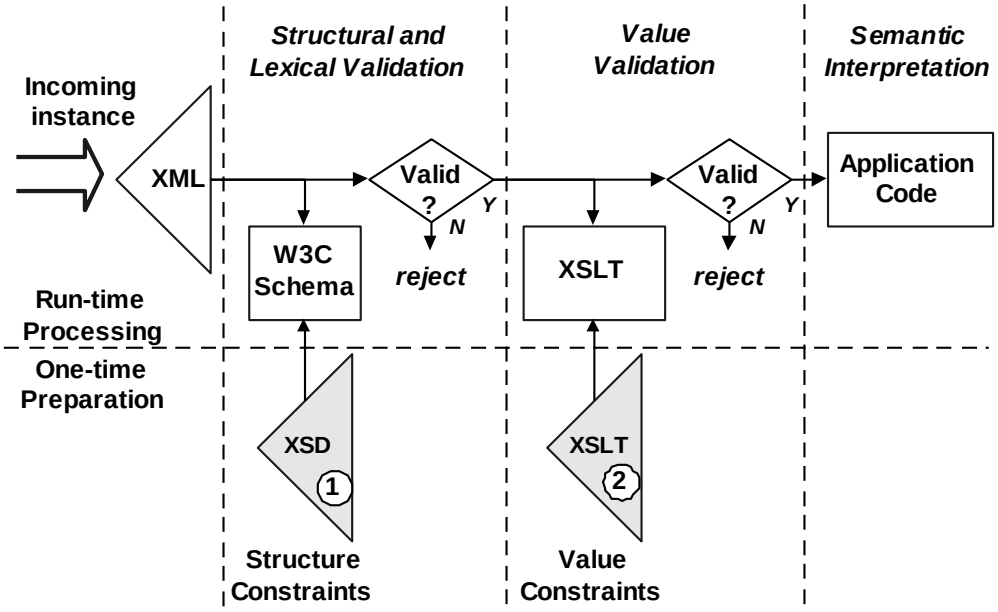


Figure 6 - Multi-phase document validation

These business rules from the second and third validation phase are formal constraints that govern the processing of a business document. Business rules are implemented as run-time assertions on the data of this business document. The outcome of a broken business rule depends on the severity of the rule.

3.1.3.1. Hard business rules

Hard business rules are constraints that generate an error. In case a business rule of this nature is broken, the processing of the message is stopped. The business document will not be sent to the final recipient. Instead, the business document state moves to error and an Application Response is generated by e-TrustEx and sent back to the sender of the message.

3.1.3.2. Soft business rules

Soft business rules are constraints that generate a warning (e.g. a specific optional field is not filled-in). In case a business rule of this nature is broken, the processing of the message is not stopped. The business document will still be sent to the final recipient. However a procedure could be implemented to notify the recipient about these warnings and in this case it is up to the recipient to decide if a manual intervention is required.

3.1.4. SOAP message

3.1.4.1. SOAP Envelope

Every document transmitted to e-TrustEx must be wrapped within a SOAP 1.1 envelope (see [REF14]). A snippet of the SOAP structure used by e-TrustEx is illustrated below:

```
<SOAP-ENV:Envelope>
    <SOAP-ENV:Header>
        - Header details -
    </SOAP-ENV:Header>
    <SOAP-ENV:Body>
        - Body details -
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The SOAP XML-based envelope carries the document in its body and defines the operation and the message being carried-out. This requirement is specified in the WSDL document.

3.1.4.2. SOAP Header

The SOAP Header consists of a custom EC Header element with a mandatory Business Header sub element. The Business Header contains the Sender Party and Receiver Party elements that are used for identifying the business partners in the transaction.

The identifiers for the business partners (Party IDs) are expressed as Identifier elements:

- A schemeID attribute defines the Issuing Agency that generates and maintains the identifier values (e.g. GLN, NAL, LEF, BE:VAT, etc. as per [REF10], [REF15] and [REF16]). If the schemeID attribute is not present, it is defaulted to GLN.
- The identifier value itself (e.g. 5790001791483, 0123456789, etc.).

The Sender ID is provided as illustrated in the following snippet:

```
<Sender>
  <Identifier schemeID="BE:VAT">0123456789</Identifier>
</Sender>
```

The Receiver ID is provided as illustrated in the following snippet:

```
<Receiver>
  <Identifier schemeID="GLN">5790001791483</Identifier>
</Receiver >
```

3.1.4.3. SOAP Body

The SOAP message body contains the actual message. The direct child element of the Body element defines the document name that is subject to the operation in the form of Submit*<Document Name>*Request. Subsequently, the direct child element of this operation element defines the message itself. This message itself is then constructed according to its proper XSD.

In the following snippet you can see a generic SOAP Body structure (in bold):

```
<Envelope>
  <Header>
    Header details
  <Header>
  <Body>
    <Submit<Document Name>Request >
      XML message itself constructed according to its proper XSD
    </Submit<Document Name>Request>
  </Body>
</Envelope>
```

Below you can see a practical example for submitting a Bundle request (SOAP body in bold):

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:oasis:names:specification:ubl:schema:xsd:CommonAggregateComponents-2"
xmlns:ec1="ec:schema:xsd:CommonAggregateComponents-2"
xmlns:urn1="urn:oasis:names:specification:ubl:schema:xsd:CommonBasicComponents-2"
xmlns:stan="http://www.unece.org/cefact/namespaces/StandardBusinessDocumentHeader"
xmlns:sac="urn:oasis:names:specification:ubl:schema:xsd:SignatureAggregateComponents-2"
xmlns:ec="ec:services:wsdl:DocumentBundle-2">
   <soapenv:Header>
      <ec:Header>
         <ec1:BusinessHeader>
            <stan:Sender>
               <stan:Identifier>TRUSTSUPPARTY1</stan:Identifier>
            </stan:Sender>
            <stan:Receiver>
               <stan:Identifier>TRUSTCUSTPARTY1</stan:Identifier>
            </stan:Receiver>
            <stan:BusinessScope>
               <stan:Scope>
                  <stan:Type>CREATE_STATUS_AVAILABLE</stan:Type>
                  <stan:InstanceIdentifier/>
               </stan:Scope>
            </stan:BusinessScope>
         </ec1:BusinessHeader>
      </ec:Header>
   </soapenv:Header>
   <soapenv:Body>
      <ec:SubmitDocumentBundleRequest>
         <ec:DocumentBundle>
            <urn1:ID>ID01</urn1:ID>
            <urn1:IssueDate>2013-09-01</urn1:IssueDate>
```

```
            <urn:SenderParty/>
            <urn:ReceiverParty/>
            <ec1:DocumentWrapperReference>
               <urn1:ID>ID01</urn1:ID>
               <urn1:DocumentTypeCode>BINARY</urn1:DocumentTypeCode>
            </ec1:DocumentWrapperReference>
         </ec:DocumentBundle>
      </ec:SubmitDocumentBundleRequest>
   </soapenv:Body>
</soapenv:Envelope>
```

3.1.4.4. SOAP Faults

SOAP Faults will be exchanged if an error occurs:

- in the Synchronous part of the Asynchronous services or
- in the Synchronous services

(Refer to section 4.2 Communication mode for a description of Synchronous and Asynchronous services.)

When the fault code is "Client", it indicates that the message could not be processed because of a problem in the SOAP Request message. In this case, the SOAP Fault object contains a Detail object that gives detailed information about the issue. The sender can use this information to correct the message and resend it to the system.

When the fault code is "Server", it means that e-TrustEx cannot process the message because of a problem in the system. In this case, the SOAP Fault object does not contain a Detail object. Resending the message immediately will not solve the issue. Instead, the sender should retry later or contact the e-TrustEx support desk to report the error and wait for a response indicating that the issue is solved.

In several specific error cases, an HTTP error instead of a SOAP Fault will be returned. This will occur when there is an issue with the request that is caught at a low level before the message processing can even begin (e.g. HTTP authentication issue or no SOAP message present in the request). The sender can use this information to correct the request and resend it to the system.

The list of SOAP faults, HTTP errors and their occurrences can be found in the annexes of this document (see [REF2]).

More information on the message processing flow and the errors returned at several steps during the flow can be found in section 3.5 Step-by-step validation flow from Sender to Receiver.

3.1.5. Binary Attachments

E-TrustEx defines two modes of sending binary attachments:

- The Submit Attached Document service where SOAP With Attachments is used. This service is used to attach small or medium size attachments to existing messages in the system.
- The Store Document Wrapper service where MTOM and HTTP chunking are used. This service is used to submit large binary files that can be referenced later by other documents. E.g. after submitting a large file using the Store Document Wrapper service, you can later submit a Bundle document that references the previously stored Document Wrapper.

3.1.5.1. SOAP with Attachments

The version in use of the WS-I Basic Profile specification does not address binary attachments. Therefore e-TrustEx implements the exchange of attachments in the context of the Submit Attached Document service as MIME encoded content (MIME part) and uses the multipart media type header at the HTTP level as a native HTTP header (see [REF17]).

A snippet of a SOAP message with MIME header at HTTP level is provided below:

```
------=_Part_0_11854491.1228781128812
Content-Type: image/jpeg
Content-ID: attached_image
```

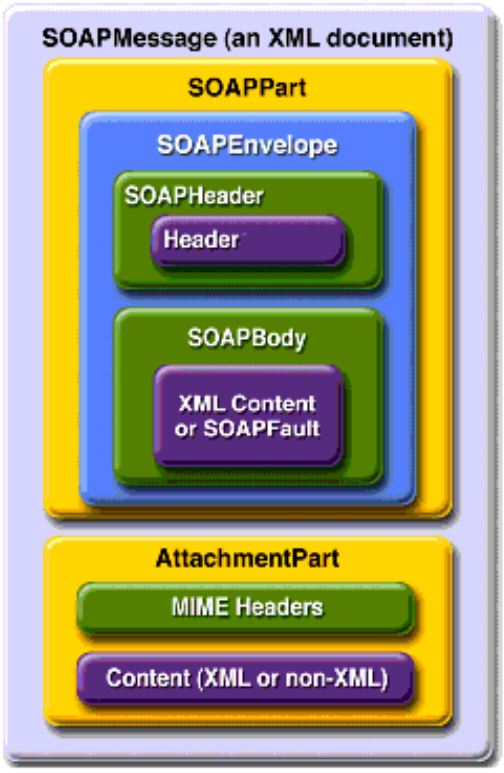


Figure 7 - SOAP message with Attachment Part

3.1.5.2. MTOM

This service slightly deviates from the other synchronous services (see 4.2.1 Synchronous Requests) offered by the platform because the Document Wrappers containing potentially large binary files are processed in two steps. The binary attachment is streamed to the file system and removed from the incoming XML message to avoid high memory consumption. The rest of the XML message is then sent to the common synchronous message handling chain of the system.

The Store Document Wrapper web service interface supports MTOM (see [REF18]) to optimize the size of the transferred encoded binary and HTTP chunking to allow the streaming of the binary file to the disk and thereby avoiding high memory consumption.

3.1.6. Code lists

Throughout the e-TrustEx messages, various XML elements contain values that are determined by code lists.

An overview per document type of the elements that use code lists can be found in the Index page of the code list document (see [REF10]).

Code list values are used in messages sent by the Sender (e.g. Bundle, AttachedDocument etc.) and in messages sent to the Sender (e.g. Status Response or Application Response).

For any messages exchanged, e-TrustEx validates via business rules that correct values for specific elements are being used (see 3.1.3 Message validation).

When a validation error occurs, depending on its severity, either a hard business rule violation or a soft business rule violation will be triggered. For instance if the XML element contains a value not present in the code list, a validation

error will occur. The exact list of business rules for each document type, containing among others these code list checks, can be found in the ICD annexes.

In the messages sent to the Parties, e-TrustEx uses code list values to return information in a short, structured and machine readable way.

E-TrustEx uses code lists from international standards organisations such as the International Organization for Standardization (ISO) or the United Nations Economic Commission for Europe (UNECE) where possible. If specific needs exist, these code lists are either customised or a new custom code list is created.

3.2. Interface definition

3.2.1. Service endpoints

E-TrustEx exposes a unified interface via web-services endpoints expressed in the Web Services Definition Language (see [REF19]). The WSDL is an XML 1.0 (see [REF20]) based document describing the specific web-service: it includes the name of the service, its location, operations and ways to communicate with it.

The effort to create an e-TrustEx web-services client should be considerably eased since each WSDL document is machine readable. To promote an interoperable exchange of messages with heterogeneous platforms (e.g. JAVA or .NET) the set of recommendations provided by WS-I Basic Profile (see [REF21]) is respected in the design of e-TrustEx web-services.

In general, one single Request-Response operation is implemented per service. This single operation receives the request and always returns a response. As it will be explained further in this document, this response is sometimes a technical acknowledgment.

In every service, e-TrustEx uses the SOAP protocol binding (see [REF14]) for requests and responses. The following must be used:

- The SOAP style attribute is document.
- The SOAP transport protocol is HTTPS.
- The SOAP operation input and output use literal style encoding.
- The SOAP action field is not used.

3.2.2. Content Based Routing

3.2.2.1. Party Identifiers

Once the service endpoint receives a request from a Sender, it is up to the pre-processing layer of e-TrustEx to handle it before delegating its processing to the back-office. For the request to be accurately routed within e-TrustEx, the Sender must include the following context information in the SOAP Header:

- The Sender ID, which identifies the Sender via a unique Party Identifier.
- The Receiver ID, which identifies the Receiver via a unique Party Identifier.

This Sender ID and Receiver ID must be expressed as Sender and Receiver elements in the SOAP Header as explained in 3.1.4.2 SOAP Header.

3.3. Security

3.3.1. Introduction

The e-TrustEx platform supports requirements for authentication, integrity and validity of documents during both transmission and storage. It also includes measures for authorization, confidentiality, auditing and non-repudiation.

These security requirements are described as follows:

- *Confidentiality* is needed to prevent third parties from eavesdropping on information that is being transmitted.
- *Authentication* ensures that the users involved in communication are really who they say they are.
- *Authorization* ensures that users only have access to the resources they are allowed to.
- *Integrity* guarantees that a message is not modified during its transmission.
- *Validity* guarantees the business validity of the messages stored in the system.

- *Auditing* controls enable relevant parties like authorized bodies to inspect transactions afterwards.
- *Non-repudiation* measures prevent users from denying actions they have undertaken.
- *Storage security* certifies that a stored message does not lose its legal attributes.

The following chapters of the document describe how the security requirements are implemented in the platform.

3.3.2. Confidentiality

The connection to the platform web services must be established using HTTPS. HTTPS (see [REF22]) is a secure version of the HTTP (see [REF23]) protocol and is being used to protect data transactions for commercial and financial purposes. It uses a Secure Socket Layer (SSL) and digital certificates to encrypt a data transfer session over an otherwise insecure HTTP connection.

The use of HTTPS guarantees transport level confidentiality however there might be a need to also ensure confidentiality at message level. The platform provides support for end-to-end encryption acting as public key repository (used to encrypt the message). However, for some business modules, the security needs do not outweigh the additional complexity of the solution and the Parties are not required to sign or encrypt the messages they send.

3.3.3. Authentication and authorization

By default, the system requires basic authentication over the HTTPS connection. The fact that HTTPS is used to set up the initial secure connection makes it difficult for someone to steal the passwords by listening on the communication as it is encrypted. The platform comes with a complete authorisation implementation based on the concept of Interchange Agreements.

A document exchange profile is a set of transactions that define operations (e.g. submission, retrieval, etc.) on specific documents.

An Interchange Agreement models the contract between parties in the context of a specific document exchange profile.

Every time a party tries to access the system, e-TrustEx checks if there is an Interchange Agreement authorising the caller to do so.

3.3.4. Integrity

The integrity of message exchange can be guaranteed by the platform through the use of XML digital signatures. The messages returned by the platform are signed and parties can be required to sign their calls to the platform web-services. This integrity is only valid for the transport level and the platform's signatures are purely technical (meaning that the business layer on top of the platform is not involved in generating or processing the signature).

The use of XML digital signatures is an optional feature and can be activated through configuration of the system. Since it adds complexity to the solution, depending on the business case, HTTPS encryption might be judged good enough to ensure the required level of integrity.

3.3.5. Validity

The platform supports multiple types of message validation: the standard XSD validation, the Schematron validation (a rule-based validation language for making assertions about the presence or absence of patterns in XML messages) and specific business rules implemented in the code of the e-TrustEx platform itself. This guarantees the business validity of the messages stored in the system.

3.3.6. Auditing

All the calls to the platform are logged and transactions are stored in the system's database. Auditors can be given temporary access to the platform database and log files in case of control.

3.3.7. Non Repudiation

When a party submits a message, e-TrustEx generates a signed acknowledgment containing information about the submitted message. This signed acknowledgment can be used as a proof of submission ensuring the non-repudiation of (submission of) messages sent through the platform. The party submitting the message is responsible for storing the acknowledgment generated by e-TrustEx.

3.3.8. Storage Security

The databases that contain business data, logging information and audit trails are kept in a secure environment to which only authorized users have access. Audit trails are kept to ensure that the data is not tampered with. Additionally, regular backups of the databases are done and securely archived.

3.4. Document Processing Considerations

3.4.1. Service Level Agreement (SLA) Rules

The e-TrustEx platform supports configuration of SLA rules that define e.g. the maximum size of binary files, the volume of messages sent in a specific timeframe or the maximum number of related documents for a message.

The SLA rules are configurable at business domain level and depending on business needs the configuration can be further restricted at transaction level. The SLA policies can be deactivated, and a flexibility marge can be added if needed.

When an SLA rule is not respected, e-TrustEx will generate a specific synchronous SOAP fault.

3.4.1.1. Maximum Message Size

Binary files attached to a message must not exceed the configured maximum message size.

3.4.1.2. Maximum Volume

A submitted binary file increases the used storage capacity that must remain under the maximum allowed during a given timeframe.

3.4.1.3. Maximum number of related messages

The number of binary files related to a message must not exceed the maximum allowed configured in the e-TrustEx platform.

3.4.2. Retention policy

The e-TrustEx platform supports the automatic removal of messages that are too old and not relevant anymore to the business through a set of retention rules.

3.4.3. Unique Message ID and Reliable Message Delivery

Reliable message delivery is required to avoid that Asynchronous Requests (e.g. Attached Documents, etc.) are processed more than once. Duplication of messages may happen due to error, multiple retries following a communication failure and similar situations. The design of the endpoints of the asynchronous services (see 4.2.2 Asynchronous Requests) aims at a situation where repeated executions of the same request have the same effect as a single execution of the request, avoiding the processing of duplicate instances of the same message. It should be noticed that Synchronous Requests are different in nature since the submission of multiple instances of the same request does not have side effects apart from the unnecessary consumption of resources.

The natural consequence, of the above, is that every Asynchronous Request should be processed once-and-once-only by e-TrustEx. This check is performed synchronously. In addition to the check concerning the uniqueness of the ID, every message will be checked regarding multiple aspects. These checks can be summed up in:

- validity of the Payload Type via the SOAP Body Wrapper element
- validity according to the message XSD and

- the uniqueness of the message ID per document type, Sender Party and Receiver Party: if the same Request is sent more than once to the same Receiver, the second attempt (and others) will generate a duplicate message ID error which will prevent that multiple instances of the same request are processed by e-TrustEx

The first point to keep in mind is that the detection of duplicate requests is made via the unique message ID (possibly combined with version ID). The second point to keep in mind is that the above steps are sequential. The next section will look closer to the message ID and its role in the reliable delivery of the message.

3.4.3.1. What is the role of the message ID?

As already discussed in this document, each WSDL will import several XML Schemas (XSDs) describing the structure of the XML documents to be exchanged by the users of a service. The message ID is found in the XSD root element of the document to be submitted by a Party in the context of an Asynchronous Request. Below is the example of a Document Bundle:

```
<Envelope>
    <Header>
        - Header details -
    </Header>
    <Body>
        <SubmitDocumentBundleRequest>
            <DocumentBundle> ←XSD root element
                …
                <ID>ID01</ID>
                <IssueDate>2007-01-08</IssueDate>
                …
            </DocumentBundle>
        </SubmitDocumentBundleRequest>
    </Body>
</Envelope>
```

In addition to the Document ID (or message ID), there is sometimes a Parent ID (e.g. in the Attached Document) which is also subject to some restrictions.

Putting it all together, this means that:

1) The ID is part of the payload of the message provided by the Sender i.e. the Sender manages the IDs of the message, not e-TrustEx.

2) The structure of the ID is the Sender's responsibility. The following rules are imposed:

   - Any blanks (spaces or tabs) in the right or left side of the ID will be trimmed so that the instances below are understood as the same one:

```
1
 1
  1
```

   - Only characters from the 7-bit ASCII table will be allowed in the message ID to prevent any issues with special characters. This full list of allowed characters is as follows:

Alphabetic characters	ABCDEFGHIJKLMNOPQRSTUVWXYZ

	abcdefghijklmnopqrstuvwxyz
Numeric characters	0123456789
Other characters	!"#$%&'()*+,-./:;<=>?@ [\]^_`{|}~

Message IDs which contain invalid characters, such as shown in the below examples, will not be accepted by e-TrustEx.

<ID>FACTURE N° 575197</ID>	(because of °)
<ID>Facture 9300015023 - Référence 7052</ID>	(because of é)
<ID>2ième trim 05</ID>	(because of è)

3) The Document ID must be unique per document type, Sender, and Receiver combination. Different types of documents can be submitted with the exact same ID.

4) The ID of a specific document type cannot be resubmitted by the same Sender to the same Receiver. Should a message of a specific document type be submitted then any retry attempted on a message of the same document type using the same message ID and Receiver will lead to a synchronous SOAP Fault.

   - Should a Sender Party submit the same ID of a specific document type to more than one Receiver (e.g. different DGs within the European Commission), then all will succeed and be put in the state received.
   - Should a Sender Party submit the same ID of a specific document type multiple times to the same Receiver, then only the first one will succeed and be put in the State Received. The others will lead to an Error.

5) The message ID will be the surrogate for any informational service on the Asynchronous Request submitted by a Party. This can be e.g. a Status Request or Retrieve Request.

3.4.4. Transactions Atomicity

Each Request resulting in an asynchronous processing is understood as an Isolated Transaction for e-TrustEx. However, in case the document requires the existence of a parent document (e.g. Attached Document requires a parent document such as an Invoice) then this request depends on a preceding event: the successful receipt of the parent document. To decrease the likelihood of rejection, e-TrustEx implements an internal retrial process. This is transparent to the Sender and aims at resolving:

- Possible Message inversion issues (e.g. a wrong message sequence in case the Sender submits a batch of messages);
- Possible delay in the delivery of one of the messages (e.g. in case the Sender submits a very large message);

3.5. Step-by-step validation flow from Sender to Receiver

The figure below illustrates the step-by-step validation flow for messages submitted by a Sender to a Receiver.

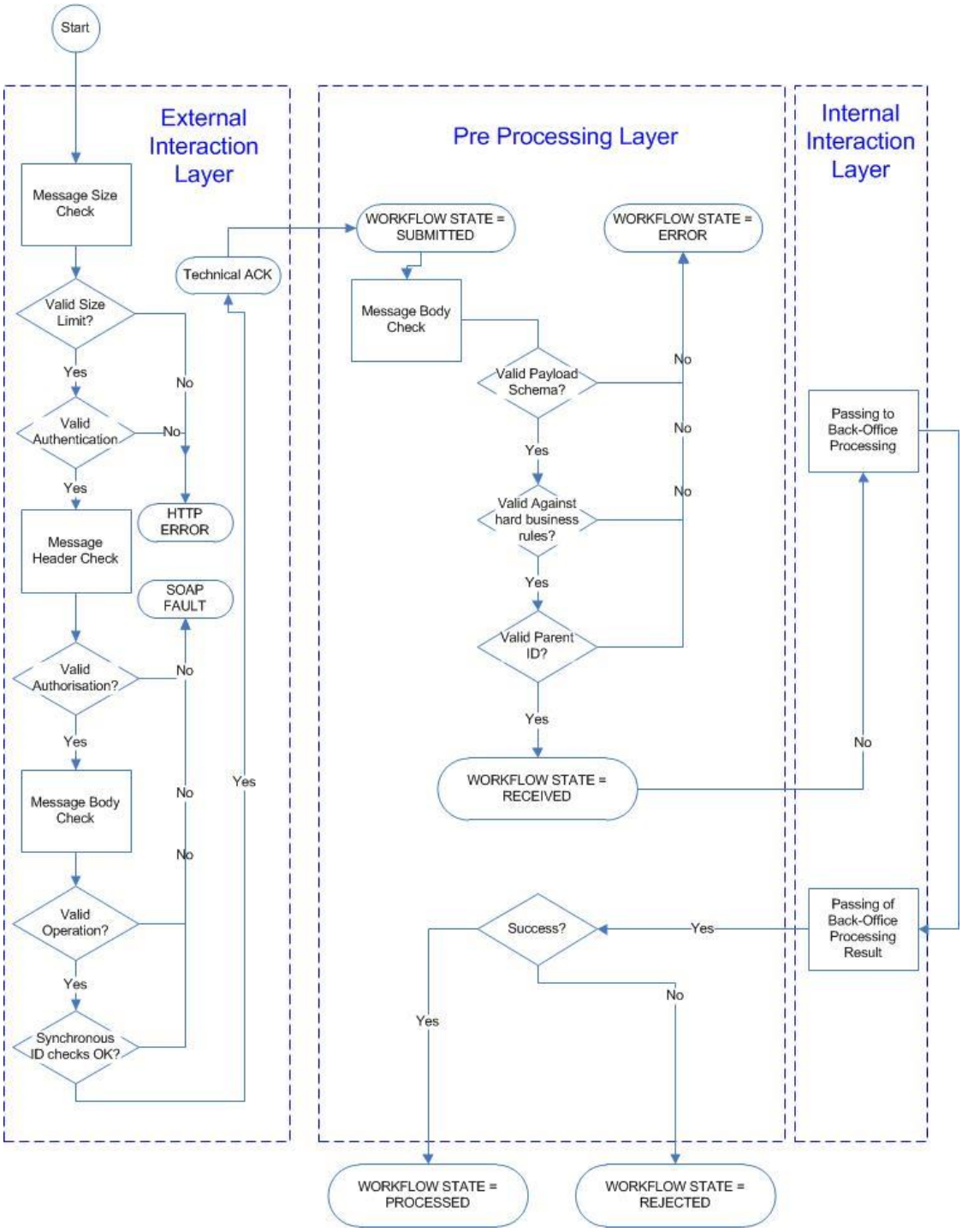


Figure 8 - Step-by-step validation

3.5.1. External Interaction Layer

In the External Interaction Layer, the Sender establishes a connection with the web-services exposed by e-TrustEx to submit a business document.

E-TrustEx will check the message size, authenticate and authorize the Sender and perform several high-level checks. It checks the message ID for its contents and length. Additionally, it checks if the message ID is unique for this Sender, Receiver and the document type sent.

If a check in the External Interaction Layer fails, the Sender receives either an HTTP error or a SOAP Fault synchronously. The connection is closed and the message will not be further processed. This implies also that no error message is created inside e-TrustEx.

If the checks in the External Interaction Layer succeed, the Sender receives a technical acknowledgment synchronously.

This technical acknowledgment only indicates that the message the Sender has submitted, has been received by e-TrustEx and that it will be further processed. It does not say that the document is valid from a technical or business perspective as only in the next layer (Pre Processing Layer) the structure and automatic business rules are checked and only in the back-office of the Receiver the message is further processed.

When the Sender receives this technical acknowledgment synchronously, the connection is closed and any further response (positive or negative) will be provided to the Sender in an asynchronous way using an Application Response.

For asynchronous messages this sums up in a first message state: SUBMITTED.

3.5.2. Pre Processing Layer

In the Pre Processing Layer, e-TrustEx performs the following operations:

- It checks the structure against the format defined in the specifications (XSD schema validation). If this operation fails, the message moves to an error state and an Application Response is sent back to the Sender.
- It checks the message against the hard business rules defined in the specifications. If this operation fails, the message moves to an error state and an Application Response is sent back to the Sender.
- It checks the message against the soft business rules defined in the specifications. If this operation fails, the message does not move to an error state and the processing continues to the next step, but a warning is generated and sent to the back-office of the Receiver together with the message. In the back-office of the Receiver the message is further processed, following the business specific workflow.
- It checks if the message contains a child-parent relationship (e.g. an Attached Document linking to an Invoice). If this operation fails, the message moves to an error state and an Application Response is sent back to the Sender. If this operation succeeds, the child-parent relationship is stored in e-TrustEx. This check takes into account the internal retrial policy described in the section 3.4.4 Transactions Atomicity.
- Optionally and depending on the document type and business domain, a human readable file could be generated (as a PDF file, html etc.) using the XML data the Sender has submitted and extra information added by the system (e.g. reception date).

If during one of these operations the message moves to an ERROR state, the processing of the message will stop without the involvement of the Internal Interaction Layer or the specific back-office processing. The Sender is then notified asynchronously of this error state by an Application Response containing the error information.

Using this information, the Sender can correct the message and resubmit it to e-TrustEx. Note that since the message has been stored in an error state in e-TrustEx, the message ID cannot be re-used and the corrected message should be sent with a new ID.

The Receiver will not be notified of this error, will not see the original message itself and will not need to take any action since the document was never correctly received by e-TrustEx (e.g. this resembles receiving a fax message which should contain an Invoice but contains only unreadable characters instead).

If the message state does not move to an ERROR state during these operations, then e-TrustEx puts it into the state RECEIVED. The moment when this state transition occurs is considered as the reception date of the message by e-TrustEx. Subsequently, based on the Receiver identification that the Sender provided, e-TrustEx will make this document available to the correct back-office for further processing.

3.5.3. Internal Interaction Layer

In the Internal Interaction Layer, the document is transferred to the Receiver.

Since e-TrustEx already has put the document state to "Received", no acknowledgment is sent when the message has been passed to the receiver.

At this point the receiver can further process the document depending on the specificities of the business domain and pass back the result via an Application Response.

When e-TrustEx receives the result, it will update the state of the document and communicate it back to the Sender (by forwarding the Application Response to the Sender, indicating that the message has been PROCESSED, REJECTED, etc. depending on the state machine configured for that document).

4. CONNECTING TO E-TRUSTEX

4.1. Parties and Agreements

4.1.1. Party

A Party is an entity that can exchange messages in e-TrustEx. Different party roles exist depending on the context. Frequently occurring party roles within e-TrustEx are BundleExchanger, Supplier and Customer. A Party has a name and a Party Identifier (or optionally a set of Party Identifiers) as defined in 3.1.4.2 SOAP Header and 3.2.2.1 Party Identifiers. It may also have credentials associated in order to authenticate itself to e-TrustEx.

We distinguish three categories of parties: Issuer Parties, Sender Parties and Receiver Parties. Note that a party could belong to all three categories depending on the actions it is undertaking.

4.1.1.1. Sender Party

A Sender Party is the official sender of a document in e-TrustEx. If the Sender Party will directly connect to the platform, it needs to have associated credentials.

4.1.1.2. Receiver Party

A Receiver Party represents the final recipient of a document sent via e-TrustEx.

4.1.1.3. Third Party

A Third Party is a party that can send messages on behalf of other parties, if the agreements allowing it exist. This type of party should always have associated credentials in order to authenticate itself to e-TrustEx.

Third parties e.g.: a sender's back-office or a service provider.

4.1.1.4. Issuer Party

An Issuer Party represents a party that uses its associated credentials to connect to the System in order to use its services for exchanging documents (it can be sometimes referred to as the caller or the requester). The credentials username and password are used to authenticate the Issuer Party using HTTP basic authentication (see [REF24]).

The Issuer Party can be the Sender Party, or a Third Party delegated by the Sender Party to submit documents in its name.

4.1.2. Agreements

4.1.2.1. Party Agreement

Each Third Party can represent one or multiple Parties through a Party Agreement.

If a Party directly connects his back-office to e-TrustEx, then this back-office connection is determined by either

- The Credentials associated to the Party. In this case the Issuer Party is the same with the Sender Party and no Party Agreement is needed.
- Or, the Credentials associated to a party representing the party's back-office that can be linked to one or more legal entities through Party Agreements. In this case the Issuer Party (the back-office) is not the same with the Sender Party (the legal entity) and a Party Agreement is needed in order to authorize the Issuer to submit documents on behalf of the Sender. E.g. an international organisation that has separate legal entities across different countries, but that share the same back-office can connect by using the same Credentials that represents all legal entities.

If a Party uses a third party service provider, then this service provider will typically have one connection to e-TrustEx (determined by its associated credentials) through which it can send messages on behalf of multiple separate Parties. The links between the third party service provider and the Parties are defined in Party Agreements.

An HTTP user and password combination (i.e. the back-office's or service provider's credentials) together with the Sender ID from the SOAP Header defines the Party Agreement.

To summarize, the party agreement defines that a specific Issuer Party can send messages on behalf of a specific Sender Party. E.g. Party X can send specific documents on behalf of Party Y, if there is a party agreement between Party Y, authorizing Party X on those specific documents.

4.1.2.2. Interchange Agreement

The Interchange Agreement is the result of adding the notions of Receiver and Profile to the above mentioned concept. The Interchange Agreement defines that a specific Sender Party can submit a specific set of documents (defined by the Profile) to a specific Receiver.

E.g. Party Y can send documents to Party Z, if there is an Interchange Agreement between Party Y and Party Z on a Profile allowing the transmission of those documents. In addition if there is a party agreement between Party Y authorizing Party X on these documents, than Party X can also send those documents to Party Z.

The Sender's ID and the Receiver's ID from the SOAP Header, as well as the submitted document/transaction define the Interchange Agreement.

4.1.3. Practical use

If the Party chooses to connect directly without a third party, the connection typically looks like in the figure below:

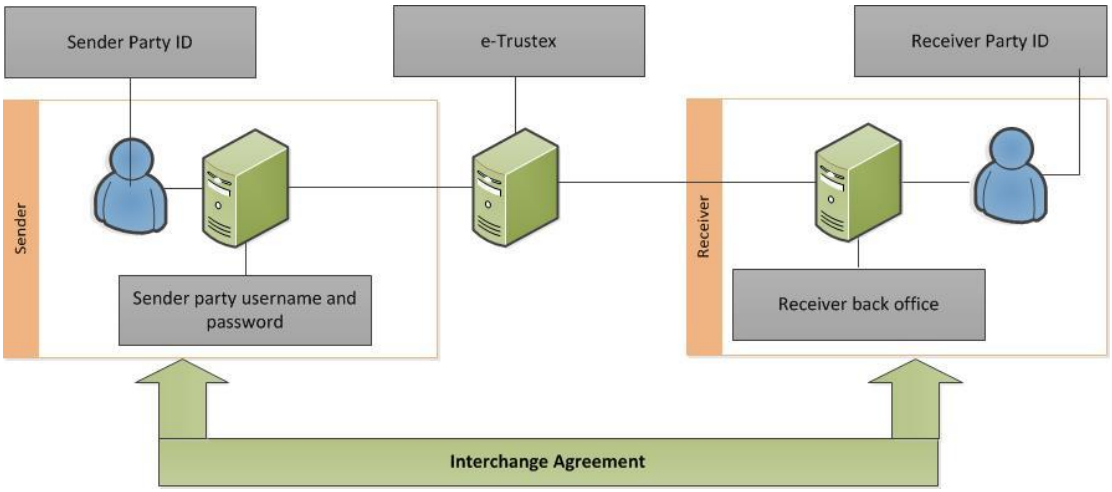


Figure 9 - Direct Party Connection

If the Party chooses to connect via a third party, the connection typically looks like in the figure below:

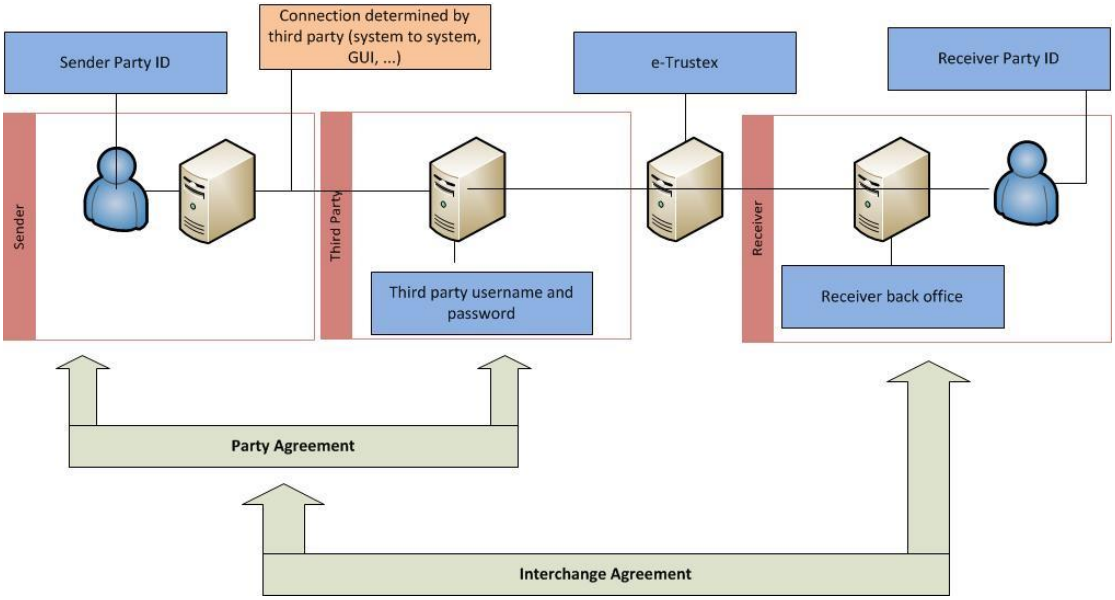


Figure 10 - Connection via Third Party

4.2. Communication mode

The following logical categorisations are used for the type of requests in e-TrustEx:

- Synchronous Requests.
- Asynchronous Requests.

Communication mode	Functional description	Technical protocol	Type of services
Synchronous	Allows the Issuer to quickly know the result of Read and Operational Requests requiring fast processing and where e-TrustEx passes the results to the requester without delay.	Request – Response SOAP web service over HTTP	Read & Operational Services
Asynchronous	It is used for Requests for which e-TrustEx needs a significant processing time or which require workflow steps performed by the back-office. During this period, the Issuer can do read requests to periodically check the message status.	Request – Response SOAP web service over HTTP	Asynchronous Services

4.2.1. Synchronous Requests

With this type of request, an immediate answer from e-TrustEx is expected.

- In the case of success, the expected response is returned to the requester. E.g. a Party submits a Status Request for a document and immediately receives a Status Response containing the status details of the document.

- In the case of an error, a SOAP Fault or HTTP error is returned as specified in 3.1.4.4 SOAP Faults and the processing stops. E.g. a third party attempts to perform a Status Request for a Party with which it does not have a Party Agreement. In this case, a SOAP Fault Unauthorized Access will be returned.

Read requests are implemented as synchronous processes and are used to get information from e-TrustEx without modifying the messages themselves. E.g. Inbox Request and Status Request.

Operational requests are also implemented as synchronous processes and are used to perform configurations or store binary files before sending the main message referring them. E.g. Create Party Request, Create Interchange Agreement Request and Store Document Wrapper Request.

A defining characteristic is that in order to get access to the Synchronous Requests, only the Issuer credentials and Party Agreement information is required. This means that in the SOAP Header only the Sender ID needs to be present. The Receiver ID, which is the defining additional parameter for the Interchange Agreement, does not need to be specified in this case.

4.2.2. Asynchronous Requests

Asynchronous Requests are services characterized by a two-step processing: a synchronous processing followed by an asynchronous one.

- If an error occurs in the synchronous processing, a SOAP Fault or HTTP error is returned synchronously as specified in 3.1.4.4 SOAP Faults and the processing stops. The message is not stored nor delivered to the recipient.
- If the synchronous processing completes successfully, a Technical Acknowledgment is returned synchronously and the message is temporarily put in SUBMITTED state while several asynchronous checks are performed. The HTTP connection is terminated and the processing of the message continues in the asynchronous part.
- If an error occurs in the asynchronous processing, the message is stored in ERROR state and an Application Response is generated. The message will not be further processed, nor delivered to the Recipient and the Sender can retrieve the Application Response from e-TrustEx to know more details on the error cause.
- If the asynchronous checks (see Figure 8 - Step-by-step validation) complete successfully, the message is stored in RECEIVED state in e-TrustEx. The further processing of the message depends on the transaction and back-office involved. E.g. in the case of Bundle, the back-office of the Receiver will pick up the processing of the message and eventually update the state of the Bundle to READ or FAILED using an Application Response. The Sender can retrieve the Application Response from e-TrustEx to know more details of the processing.

For Asynchronous Requests an Interchange Agreement is required. Therefore, both the Sender ID and the Receiver ID from the SOAP Header are required.

4.2.3. Technical Acknowledgment

A Technical Acknowledgment (Ack) is returned by e-TrustEx in the case of Asynchronous requests, in the end of the synchronous part. In this case, it means that the message has been received and the processing of the message continues in the asynchronous part.

The technical acknowledgment is digitally signed and must be saved and archived by the sender if proof of the message reception is required. Without this proof a sender cannot claim to have sent a message if it is not present in e-TrustEx (e.g. due to technical issues in the connection between the sender and e-TrustEx).

Table 1 - Acknowledgement

Element	Description
SOAP Header	
BinarySecurityToken	Security token that is binary encoded

XMLDSIG Signature	<Signature ID?> <SignedInfo> <CanonicalizationMethod/> <SignatureMethod/> (<Reference URI? > (<Transforms>)? <DigestMethod> <DigestValue </Reference>)+ </SignedInfo> <SignatureValue> (<KeyInfo>)? (<Object ID?>)* </Signature>
SOAP BODY	
AckIndicator	*AckIndicator – Boolean value*
IssueDate	*The IssueDate is the system date of reception.*
ID of the referenced message	*AcknowledgedDocumentReference.DocumentReference.ID*
DocumentTypeCode of the referenced message	*AcknowledgedDocumentReference.DocumentReference.DocumentTypeCode (E.g. "380" for Invoice, "BDL" for Bundle etc.)*
SenderPartyID	*AcknowledgedDocumentReference.DocumentReference.SenderParty.EndpointID – the Sender Party ID from the business header of the message*
ReceiverPartyID (optional)	*AcknowledgedDocumentReference.DocumentReference.ReceiverParty.EndpointID - the Receiver Party ID from the business header of the message (if present in the request)*

4.2.4. Notification Available

For the asynchronous services, E-TrustEx checks if an availability notification is required, and if it is the case it generates an application response providing the document type code and having response code 1 and makes it available to the sender party. This way the sender is aware of the message state change to RECEIVED, marking the fact that the message has been correctly processed by eTrustEx and it is ready to be retrieved by or dispatched to the recipient(s).

4.2.5. The Send/Receive interaction scenario

The following scenario depicts the most common use of e-TrustEx by both Senders and Receivers.

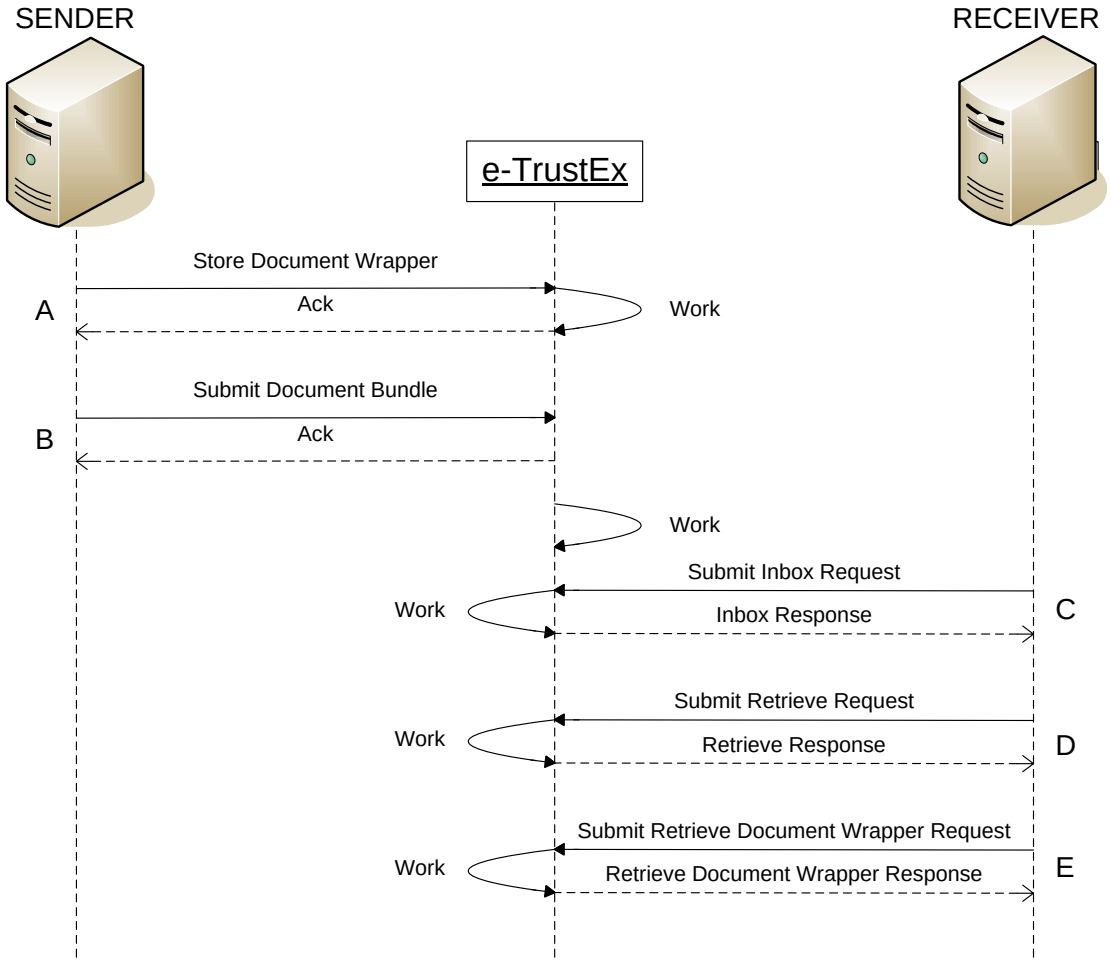


Figure 11 – Send/Receive Interaction Scenario

Table 2 - Send/Receive Interaction Scenario

Step ID	Description	Communication mode
A	The Sender submits a Document Wrapper containing the binary file to be further attached and sent via a Document Bundle. E-TrustEx responds with an Acknowledgment meaning that the Document Wrapper has been successfully stored onto the server. This step can be repeated as many times as necessary since the Document Bundle can reference up to 1500 Document Wrappers, depending on the business domain.	Synchronous
B	The Sender prepares and submits a Document Bundle which encloses the reference(s) to the previously sent Document Wrapper(s). E-TrustEx responds synchronously with a technical Acknowledgment (Ack) meaning that the message is accepted for processing, and closes the connection. E-TrustEx continues to process the message asynchronously.	Asynchronous

Step ID	Description	Communication mode
C	The Receiver checks its Inbox via an Inbox Request. E-TrustEx synchronously responds with the Inbox Response containing all documents addressed to the Receiver and still not retrieved by the Receiver.	Synchronous
D	The Receiver retrieves the received Document Bundle by invoking the Retrieve service. The references to the attached binary files are found in the Document Bundle and shall be used to perform the next step.	Synchronous
E	The Receiver can download the files that have been received as part of the Document Bundle by invoking the Retrieve Document Wrapper service for each of the binaries referenced thereby.	Synchronous

Section 4.3 Services describes the processing of messages in more details for each of the services mentioned in the above table.

4.2.6. Notification and "Store-and-Forward"

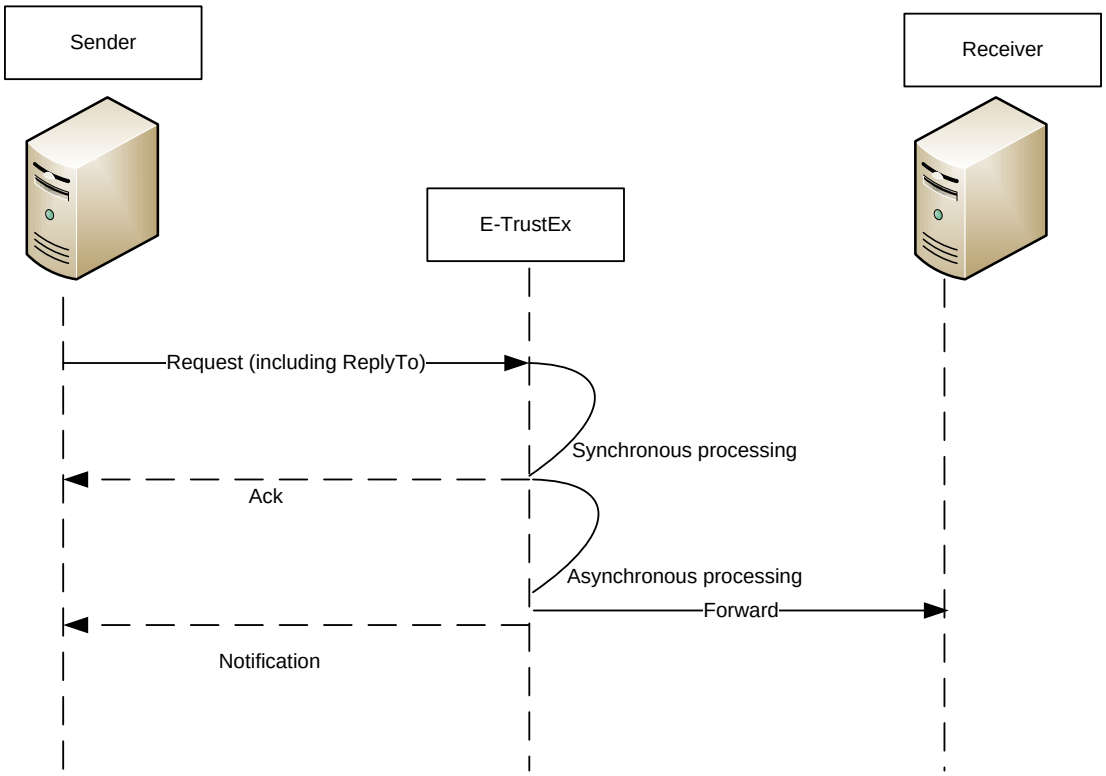


Figure 12 - Notification and Store-and-Forward

4.2.6.1. "Store-and-Forward"

In case of store-and-forward communication pattern, e-TrustEx will dispatch the business documents to the recipient back-offices using standard interfaces (JMS, Web Service, AMQP). Metadata about the back-office interfaces is configurable at database level.

4.2.6.2. Notification

In case the Sender would like to be notified at the end of the asynchronous part of the processing, e-TrustEx will inform the Sender on the status of the processing. See also 4.2.4 Notification Available.

4.3. Services

4.3.1. Profiles

The interaction rules between the Parties and e-TrustEx must be defined in a way that fits the capabilities of a vast number of Parties of different sizes and business. E-TrustEx enables Senders to choose from different Profiles which describe the capabilities to be supported by the Sender.

E-TrustEx exposes a Generic Profile to the Parties, called Bundle. Other profiles can be configured depending on the business needs.

4.3.2. Service list

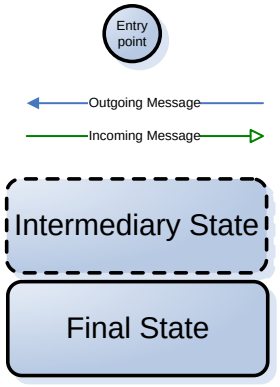
The table below provides a list of services provided by e-TrustEx.

Table 3 e-TrustEx services

Service Type	Service Name	Communication mode	Profiles
Write	Submit Document Bundle	Asynchronous	Bundle
	Submit Application Response	Asynchronous	Bundle
	Submit Attached Document	Asynchronous	Bundle
	Submit Event Notification	Asynchronous	Bundle
Read	Retrieve Document Wrapper	Synchronous	Bundle
	Inbox Request	Synchronous	Bundle
	Retrieve Request	Synchronous	Bundle
	Query Request	Synchronous	Bundle
	View Request	Synchronous	Bundle
	Status Request	Synchronous	Bundle
	Retrieve Interchange Agreements	Synchronous	Bundle
Operational	Store Document Wrapper	Synchronous	Bundle
	Delete Document Wrapper	Synchronous	Bundle
	Delete Document	Synchronous	Bundle
	Create Party	Synchronous	Admin
	Create Interchange Agreement	Synchronous	Admin

As observed in the above table, all services related to documents exchange are found in the generic business domain under the generic profile called Bundle. They can be configured at any time by adding them to a new profile and/or business domain, depending on the business needs. Additionally there are two operational services that allow for the configuration of new parties and interchange agreements. These services are gathered under Admin profile.

The following symbols are being used in the workflows depicted in the next sections:



4.3.3. Asynchronous/Write services

4.3.3.1. Document Bundle

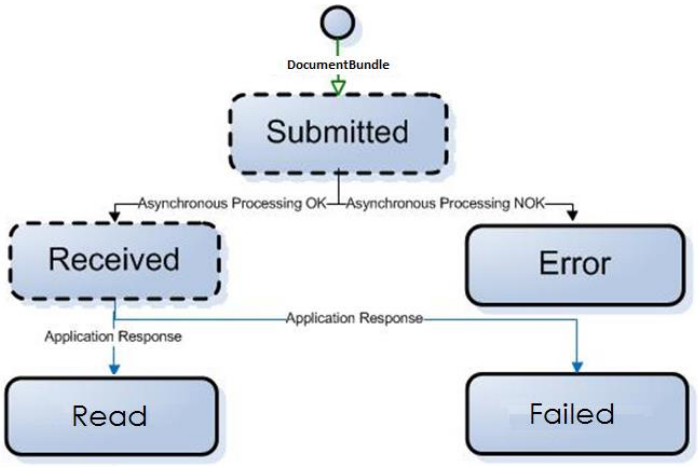


Figure 13 – Document Bundle communication workflow

Trigger: Submission of a Document Bundle by a Party		
Layer where validations are performed	**In case of Success**	**In case of Failure**
External Interaction Layer	Technical Ack	HTTP error or SOAP Fault
Pre-Processing Layer	State is set to "RECEIVED" + An Application Response with Response Code and Description is sent by e-TrustEx to the Party: • BDL:1 in case available notification is needed	State is set to “ERROR” + An Application Response with Response Code (as presented in the list below) and Description containing error details is sent by e-TrustEx to the Party: • BDL:4 in case of hard business rule violation or SLA policy violation • BDL:5 in case of non-existing parent • BDL:6 in case of a non-existing document wrapper referenced in the message
Back-office Processing	An Application Response with Response Code BDL:7 is sent by the Receiver to the Sender. + State is changed to "READ"	An Application Response with Response Code BDL:4 is sent by the Receiver to the Sender. + State is changed to "FAILED"

Implementation notes:

- Each Bundle Document may reference zero or more document wrappers.
- A Bundle Document can be sent at any point in time by a Party after the parent document and the referenced wrapper documents have been received by e-TrustEx.
    - Sending a Bundle Document after the parent document has been processed does not make sense since the parent document has reached a final state. This Bundle Document will not be handled anymore by the Receiver's back-office.
    - E-TrustEx has not implemented a retrial policy in case of the parent document not found. This means that the message will have the state changed to ERROR from the first failed attempt to identify the parent document and an Application Response will be generated by e-TrustEx. Sending the parent document after this event, will not cause the Bundle Document to be processed anymore, it will remain in Error state. The Sender should now submit a new Bundle Document message with a Document ID that has not been used before.
    - Sending a Bundle Document before the referenced wrapper documents have been processed does not make sense since they will not be found and the message will change state to ERROR, which is a final state. If this happens the Sender will need to resubmit the bundle with a new unique document ID.

4.3.3.2. Application Response

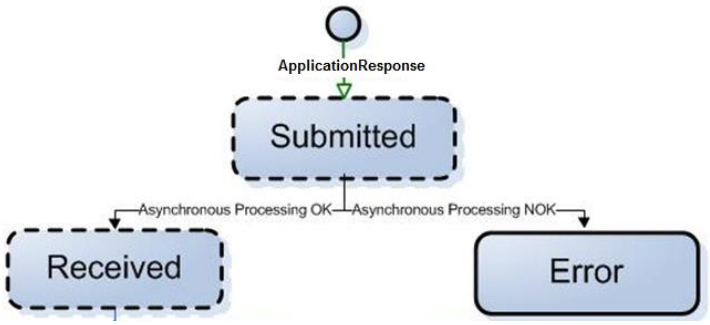


Figure 14 – Application Response communication workflow

Trigger: Submission of an Application Response by a Party		
Layer where validations are performed	**In case of Success**	**In case of Failure**
External Interaction Layer	Technical Ack	HTTP error or SOAP Fault
Pre-Processing Layer	State is set to "RECEIVED"	State is set to “ERROR” + An Application Response with Response Code (as presented in the list below) and Description containing error details is sent by e-TrustEx to the Party: • 301:4 in case of hard business rule violation • 301:5 in case of non-existing parent • 301:6 in case the parent document is not in the correct state
Back-office Processing	N/A	N/A

Practical use:

- Notifies the caller of an error occurred in the pre-processing layer (the asynchronous processing) and it is sent by the System
- Triggers the message state change when sent by the Receiver of the message
- Notifies the Sender of the availability of the message if an availability notification is required, providing the document type code and having response code 1

4.3.3.3. Attached Document

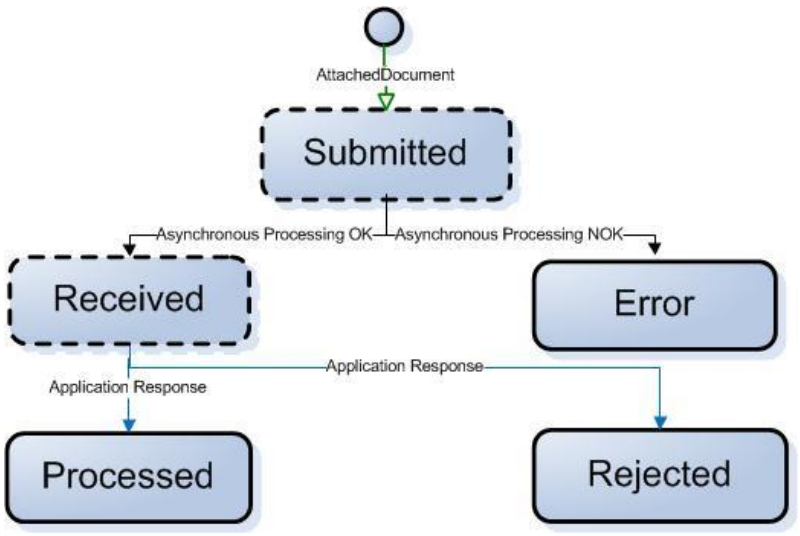


Figure 15 - Attached Document communication workflow

Trigger: Submission of an Attached Document by a Party		
Layer where validations are performed	**In case of Success**	**In case of Failure**
External Interaction Layer	Technical Ack	HTTP error or SOAP Fault
Pre-Processing Layer	State is set to "RECEIVED"	State is set to “ERROR” + An Application Response with Response Code (as presented in the list below) and Description containing error details is sent by e-TrustEx to the Party: • 916:4 in case of hard business rule violation • 916:5 in case of non-existing parent
Back-office Processing	An Application Response with Response Code 916:1 is sent by the Receiver to the Sender. + State is changed to "PROCESSED"	Application Response with Response Code 916:2 is sent by the Receiver to the Sender. + State is changed to "REJECTED"

Implementation notes:

- The processing of an Attached Document will usually occur automatically by the back-office based on the MIME type. This MIME type can be retrieved from the HTTP headers in the RetrieveRequest Response of an AttachedDocument type and it is found in the header Content-Type of the part corresponding to the attached binary.

- Each Attached Document will only contain 1 binary file, but multiple Attached Documents can be linked to the same parent document.
- An Attached Document can be sent at any point in time by a Party after the parent document has been received by e-TrustEx.
    - If a Receiver informs the Sender that a document is missing for the parent document that needs to be processed, the Sender can still submit it at that moment without the need of changing the state of the parent document. After the requested Attached Document is received, the processing of the parent document can continue in the Receiver's back-office.
    - Sending an Attached Document after the parent document has been processed does not make sense since the parent document has reached a final state. These Attached Documents will not be handled anymore by the concerned Agent in the Receiver's back-office.
    - E-TrustEx has implemented a waiting room for Attached Documents that accidentally are received before the parent document is received. This means that e-TrustEx will retry several times in a specific timeframe to find the parent of the Attached Document. If the retry attempts exceed a specific number, then the Attached Document will transfer to the Error state and an Application Response will be generated by e-TrustEx. Sending the parent document after this event, will not cause the Attached Document to be processed anymore, it will remain in Error state. The Sender should now submit a new Attached Document message with a Document ID that has not been used before.

4.3.3.4. Event Notification

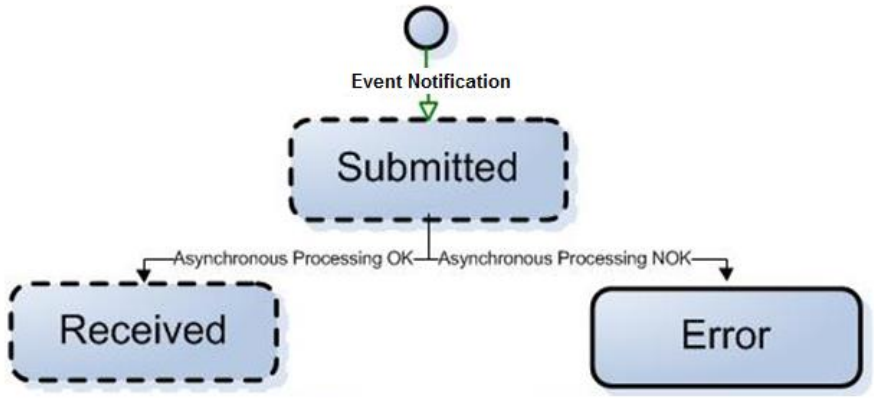


Figure 16 – Event Notification communication workflow

Trigger: Submission of an Event Notification by the System		
Layer where validations are performed	**In case of Success**	**In case of Failure**
External Interaction Layer	Technical Ack	HTTP error or SOAP Fault
Pre-Processing Layer	State is set to "RECEIVED"	State is set to “ERROR” + An Application Response with Response Code (as presented in the list below) and Description containing error details is sent by e-TrustEx to the Party: • EVN:4 in case of hard business rule violation
Back-office Processing	N/A	N/A

Practical use:

- Notifies the concerned parties of modifications regarding party and interchange agreements configurations that occurred in their business domain.
- Notifies concerned parties of document deletions that occurred in their business domain.

These parties need to be configured in e-TrustEx in order to receive such notifications.

4.3.4. Synchronous Services

4.3.4.1. Read Services

4.3.4.1.1. Retrieve Document Wrapper

Trigger: Submission of a Retrieve Document Wrapper Request by a Party		
Validations performed at the:	**In case of Success**	**In case of Failure**
External Interaction Layer	Retrieve Document Wrapper Response	HTTP error or SOAP Fault
Pre-Processing Layer	N/A	N/A
Back-office Processing	N/A	N/A

The processing of a Retrieve Document Wrapper request is fully synchronous and should be considered as a single, atomic step.

The essential parameters for the system to perform the Retrieve Document Wrapper Request are provided below. They are all found in the message.

Element	**Location**	**Description**
Request Sender Party Endpoint ID	*Header.BusinessHeader.Sender.Identifier*	The Request Sender Party: it can be the Sender or the Receiver of the message to retrieve
ID	*DocumentReferenceRequest.ID*	The ID of the document to retrieve

Document Type Code	*DocumentReferenceRequest.DocumentTypeCode*	The type code of the document to retrieve
Sender Party ID	*SenderParty.EndpointID*	The Sender Party of the document to retrieve

At the end of the processing, the System returns a RetrieveDocumentWrapperResponse message having:

- in its document content the data of the requested document wrapper and
- the binary content within a MIME multipart

There should be a single document wrapper corresponding to the specified criteria. If none or more than one document wrapper is found for the specified criteria, then the document content in the response message will be empty.

The Retrieve Document Wrapper Request must respect the following Business Rules. If one of these rules is not respected, a SOAP Fault is returned synchronously instead of the RetrieveDocumentWrapperResponse message.

Business Rule	**Severity**
The Sender Party ID must be present and not empty in the message payload	*Hard*

4.3.4.1.2. Inbox Request

Trigger: Submission of an Inbox Request by a Party		
Validations performed at the:	**In case of Success**	**In case of Failure**
External Interaction Layer	Inbox Response	HTTP error or SOAP Fault
Pre-Processing Layer	N/A	N/A
Back-office Processing	N/A	N/A

The essential parameters for the system to perform the Inbox Request are all found in the SOAP Header. No parameters are present in the message.

The System queries for non-retrieved documents of all types (except document wrappers), less than 1-year old, having as Receiver the Sender of the request, pertaining to a transaction set granted to the Issuer by an eventual party agreement with the Sender. How messages can be marked as retrieved is explained in section 4.3.4.1.3 Retrieve Request. The use of only the Inbox Request service does not mark the messages as retrieved.

It is also possible to build the response with only the documents for which the Issuer of the parent document is the same as the request Issuer and that fulfil the above criteria. In order to do this, a specific configuration must be set up on the business domain of the Issuer.

With the result of the query, the system builds the Inbox Response. The documents are ranked by received date descending. The documents having the same received date are ordered at random. Note that maximum 500 messages are returned in the Inbox Response, so it is required to mark messages as read using the Retrieve Request to remove messages that are processed by the Requester's System from responses to future Inbox Requests.

The following data is returned on the set of matching documents (1 or more):

Element	**Location**
Response Code (applicable for a response message only)	*DocumentReferenceResponseParentDocs.DocumentReferenceResponse.Response.ResponseCode*
Document Status Code	*DocumentReferenceResponseParentDocs.DocumentReferenceResponse.Status.StatusReasonCode*

ID (of the document)	*DocumentReferenceResponseParentDocs.DocumentReferenceResponse.DocumentReference.ID*
Issue Date (of the document)	*DocumentReferenceResponseParentDocs.DocumentReferenceResponse.DocumentReference.IssueDate*
Document Type Code	*DocumentReferenceResponseParentDocs.DocumentReferenceResponse.DocumentReference.DocumentTypeCode*
Sender Party Endpoint ID	***Identification value:*** *DocumentReferenceResponseParentDocs.DocumentReferenceResponse.SenderParty.EndpointID* ***Identification scheme id:*** *DocumentReferenceResponseParentDocs.DocumentReferenceResponse.SenderParty.EndpointID.@schemeID*
Sender Party Identification	**For each party identifier of the sender party:** **- the identifier value** *DocumentReferenceResponseParentDocs.DocumentReferenceResponse.SenderParty.PartyIdentification[i].ID* **- the identifier scheme id** *DocumentReferenceResponseParentDocs.DocumentReferenceResponse.SenderParty.PartyIdentification[i].ID.@schemeID*
Receiver Party Endpoint ID	***Identification value:*** *DocumentReferenceResponseParentDocs.DocumentReferenceResponse.ReceiverParty.EndpointID* ***Identification scheme id:*** *DocumentReferenceResponseParentDocs.DocumentReferenceResponse. ReceiverParty.EndpointID.@schemeID*
Receiver Party Identification	**For each party identifier of the receiver party:** **- the identifier value** *DocumentReferenceResponseParentDocs.DocumentReferenceResponse. ReceiverParty.PartyIdentification[i].ID* **- the identifier scheme id** *DocumentReferenceResponseParentDocs.DocumentReferenceResponse ReceiverParty.PartyIdentification[i].ID.@schemeID*

When information exists about a document's parent, then the following information corresponding to each parent is also returned for correlation purposes:

Element	Location
Response Code (applicable for a response message only)	*DocumentReferenceResponseParentDocs.ParentDocument.Response.ResponseCode*
Document Status Code	*DocumentReferenceResponseParentDocs.ParentDocument.Status.StatusReasonCode*
ID	*DocumentReferenceResponseParentDocs.ParentDocument.DocumentReference.ID*
Issue Date (of the	*DocumentReferenceResponseParentDocs.ParentDocument.Docum*

document)	*entReference.IssueDate*
Document Type Code	*DocumentReferenceResponseParentDocs.ParentDocument.DocumentReference.DocumentTypeCode*
Sender Party Endpoint ID	***Identification value:*** *DocumentReferenceResponseParentDocs.ParentDocument.SenderParty.EndpointID* ***Identification scheme id:*** *DocumentReferenceResponseParentDocs.ParentDocument.SenderParty.EndpointID.@schemeID*
Sender Party Identification	**For each party identifier of the sender party:** **- the identifier value** *DocumentReferenceResponseParentDocs.ParentDocument.SenderParty.PartyIdentification[i].ID* **- the identifier scheme id** *DocumentReferenceResponseParentDocs.ParentDocument.SenderParty.PartyIdentification[i].ID.@schemeID*
Receiver Party Endpoint ID	***Identification value:*** *DocumentReferenceResponseParentDocs.ParentDocument.ReceiverParty.EndpointID* ***Identification scheme id:*** *DocumentReferenceResponseParentDocs.ParentDocument.ReceiverParty.EndpointID@schemeID*
Receiver Party Identification	**For each party identifier of the receiver party:** **- the identifier value** *DocumentReferenceResponseParentDocs.ParentDocument.ReceiverParty.PartyIdentification[i].ID* **- the identifier scheme id** *DocumentReferenceResponseParentDocs.ParentDocument.ReceiverParty.PartyIdentification[i].ID@schemeID*

4.3.4.1.3. Retrieve Request

Trigger: Submission of a Retrieve Request by a Party		
Validations performed at the:	**In case of Success**	**In case of Failure**
External Interaction Layer	Retrieve Response	HTTP error or SOAP Fault
Pre-Processing Layer	N/A	N/A
Back-office Processing	N/A	N/A

The essential parameters for the system to perform the Retrieve Request of a business document are provided below. They are all found in the message.

Element	**Location**
ID (of the document)	*DocumentReferenceRequest.ID*
Document Type Code	*DocumentReferenceRequest.DocumentTypeCode*
Sender Party Endpoint	*SenderParty.EndpointID*

ID	
Receiver Party Endpoint ID	*ReceiverParty.EndpointID*
Retrieve Indicator	*RetrieveIndicator* It is a mandatory flag (Boolean type), which impacts the result of an Inbox Request by allowing the change of the retrieved flag of the business document corresponding to the search parameters. When marked as "True", the retrieved flag of the identified message is set to "TRUE", only when the request sender is the same as the message receiver. The retrieved date and time are also set to the system date and time provided the aforementioned condition is met. Messages in the System having retrieved flag marked as "True" will not be included in the result of an inbox request – see 4.3.4.1.2 Inbox Request

The Retrieve Request must also respect the following Business Rules. If one of these rules is not respected, a SOAP Fault is returned synchronously instead of the Retrieve Response.

Business Rule	Severity
The Sender Party Endpoint ID must be present and not empty in the message payload.	*Hard*
The Receiver Party Endpoint ID must be present and not empty in the message payload.	*Hard*
The Sender Party ID present in the SOAP Business header must be equal to the Sender Party Endpoint ID or the Receiver Party Endpoint ID present in the message payload. (In other words, this rule means that the Party for which the Retrieve Request is performed is either the Sender or the Receiver of the message.)	*Hard*
Document type code is mandatory.	*Hard*
Document id is mandatory.	*Hard*
Retrieve Indicator is mandatory.	*Hard*

The system queries for one document matching the request criteria. Document wrappers are not searched for by this service. In order to retrieve a document wrapper, the service Retrieve Document Wrapper should be used, as described in section 4.3.4.1.1 Retrieve Document Wrapper. If the Retrieve Indicator is set to “true” and the request sender party is the same with the message receiver party, the message is marked as retrieved and will not appear in the Inbox Response anymore (see 4.3.4.1.2 Inbox Request).

Note that setting the Retrieve Indicator to “false” when the message has already been marked as retrieved will not change the retrieve status of the message. It will remain marked as retrieved.

Based on the result of the query, the system builds the Retrieve Response message. For the document matching the criteria, the full XML content is returned in the response message.
Note that in addition to the XML content, the response message contains the Received Date (the date when the original message was received in e-TrustEx).
When retrieving an Attached Document message, the binary attachment is returned in a separate MIME part element.

4.3.4.1.4. Query Request

Trigger: Submission of a Query Request by a Party		
Validations performed at the:	**In case of Success**	**In case of Failure**
External Interaction Layer	Query Response	HTTP error or SOAP Fault
Pre-Processing Layer	N/A	N/A
Back-office Processing	N/A	N/A

The parameters for the system to perform the Query Request are detailed in the following table:

Parameter	Definitions	User Defined Value	Default Value
Period (i.e. document received date range)	Used to restrict the search to a specific time period. When the start and end date of the period are the same then the search is made on this day.	When present in the message then the difference between the *Start Date* and *End Date* must be smaller or equal to one year.	When no User Defined Value is specified in the message then the search is made on one year period prior to and including the search date.
(Set of) Document Type Codes	Used to restrict or enlarge the set of document types in the result set	When present in the message then the search is made on the set of valid document types specified in the message.	When no User Defined Value is specified then the result set includes every outbound document type.
ID of the Originator Document Reference	Used to restrict or enlarge the set of document in the result set to the ones that have a particular Originator Document Reference	When present in the message then the search is made on the set of valid Originator Document Reference.ID specified in the message.	When no User Defined Value is specified then the result set doesn't take into account the originator Document Reference.
Retrieved Documents Indicator	Boolean parameter used to enable the addition of retrieved documents into the result set.	When present in the message and set to true then retrieved documents of the specified document types or default document types are included in the message.	When no User Defined Value is specified then the retrieved documents are excluded from the result set.
Retrieved Extensions Indicator	Boolean parameter used to enable the addition of UBL extensions for all matching documents.	When present in the message and set to true then retrieved documents matching the search criteria will include the UBL extensions.	When no User Defined Value is specified then the retrieved documents do not include the UBL extensions.
(Set of) Receiver Endpoint IDs	Used to restrict the set of documents to the ones sent to a particular set of Receiver Endpoint IDs.	When present in the message then the search is made on the set of Receiver IDs specified in the message.	When no User Defined Value is specified then no filtering is applied for the message receiver.

(Set of) Sender Endpoint IDs	Used to restrict the set of documents to the ones sent by a particular set of Sender Endpoint IDs.	When present in the message then the search is made on the set of Sender Endpoint IDs specified in the message.	When no User Defined Value is specified then no filtering is applied for the message sender.

The Query Request must also respect the following Business Rules. If one of these rules is not respected, a SOAP Fault is returned synchronously instead of the Query Response.

Business Rule	Severity
When Period element is present, Period.StartDate.Date must be present	*Hard*
When Period element is present, Period.EndDate.Date must be present	*Hard*
Period. End Date. Date: The End date must be bigger or equal to the Start date	*Hard*
Period. End Date. Date: The difference between the Start date and the End date must be smaller or equal to one year	*Hard*

The documents must pertain to a transaction set granted to the Issuer by an eventual party agreement with the Sender.

The system queries for documents using the following parameters specified in the table above:

Period AND

(Set of) Document Type Codes AND

Retrieved Documents Indicator AND

WHERE

(The sender of the message is the request sender AND the receiver of the message is any of the Parties specified in the (Set of) Receiver Endpoint IDs)

OR

(The receiver of the message is the request sender AND the sender of the message is any of the Parties specified in the (Set of) Sender Endpoint IDs)

With the result of the query, the system builds the Query Response. The documents are ranked by received date descending. The documents having the same received date are ordered at random. Note that maximum 500 messages are returned in the Query Response. In the case the maximum number of messages is returned, you should refine your Query Request to ensure that all relevant messages are visible in the Query Response.

The following data is returned on the set of matching documents (1 or more):

Element	Description
Response Code (applicable for a response message only)	*DocumentReferenceResponseParentDocs.DocumentReferenceResponse.Response.ResponseCode*
Document Status Code	*DocumentReferenceResponseParentDocs.DocumentReferenceResponse.Status.StatusReasonCode*
ID (of the document)	*DocumentReferenceResponseParentDocs.DocumentReferenceResponse.DocumentReference.ID*
Issue Date (of the document)	*DocumentReferenceResponseParentDocs.DocumentReferenceResponse.DocumentReference.IssueDate*

Document Type Code	*DocumentReferenceResponseParentDocs.DocumentReferenceResponse.DocumentReference.DocumentTypeCode*
Sender Party Endpoint ID	***Identification value:*** *DocumentReferenceResponseParentDocs.DocumentReferenceResponse.SenderParty.EndpointID* ***Identification scheme id:*** *DocumentReferenceResponseParentDocs.DocumentReferenceResponse.SenderParty.EndpointID.@schemeID*
Sender Party Identification	**For each party identifier of the sender party:** **- the identifier value** *DocumentReferenceResponseParentDocs.DocumentReferenceResponse.SenderParty.PartyIdentification[i].ID* **- the identifier scheme id** *DocumentReferenceResponseParentDocs.DocumentReferenceResponse.SenderParty.PartyIdentification[i].ID.@schemeID*
Receiver Party Endpoint ID	***Identification value:*** *DocumentReferenceResponseParentDocs.DocumentReferenceResponse.ReceiverParty.EndpointID* ***Identification scheme id:*** *DocumentReferenceResponseParentDocs.DocumentReferenceResponse.SenderParty.EndpointID.@schemeID*
Receiver Party Identification	**For each party identifier of the receiver party:** **- the identifier value** *DocumentReferenceResponseParentDocs.DocumentReferenceResponse. ReceiverParty.PartyIdentification[i].ID* **- the identifier scheme id** *DocumentReferenceResponseParentDocs.DocumentReferenceResponse ReceiverParty.PartyIdentification[i]**.ID.@schemeID***

When information exists about its parent, then this information is returned for correlation purposes:

Element	Description
Response Code (applicable for a response message only)	*DocumentReferenceResponseParentDocs.ParentDocument.Response.ResponseCode*
Document Status Code	*DocumentReferenceResponseParentDocs.ParentDocument.Status.StatusReasonCode*
ID	*DocumentReferenceResponseParentDocs.ParentDocument.DocumentReference.ID*
Issue Date (of the document)	*DocumentReferenceResponseParentDocs.ParentDocument.DocumentReference.IssueDate*
Document Type Code	*DocumentReferenceResponseParentDocs.ParentDocument.DocumentReference.DocumentTypeCode*
Sender Party Endpoint	***Identification value:***

ID	*DocumentReferenceResponseParentDocs.ParentDocument.SenderParty.EndpointID* ***Identification scheme id:*** *DocumentReferenceResponseParentDocs.ParentDocument.SenderParty.EndpointID.@schemeID*
Sender Party Identification	**For each party identifier of the sender party:** **- the identifier value** *DocumentReferenceResponseParentDocs.ParentDocument.SenderParty.PartyIdentification[i].ID* **- the identifier scheme id** *DocumentReferenceResponseParentDocs.ParentDocument.SenderParty.PartyIdentification[i].ID.@schemeID*
Receiver Party Endpoint ID	***Identification value:*** *DocumentReferenceResponseParentDocs.ParentDocument.ReceiverParty.EndpointID* ***Identification scheme id:*** ***DocumentReferenceResponseParentDocs.ParentDocument.ReceiverParty.EndpointID.@schemeID***
Receiver Party Identification	**For each party identifier of the receiver party:** **- the identifier value** *DocumentReferenceResponseParentDocs.ParentDocument.ReceiverParty.PartyIdentification[i].ID* **- the identifier scheme id** *DocumentReferenceResponseParentDocs.ParentDocument.ReceiverParty.PartyIdentification[i]****.ID.@schemeID****

4.3.4.1.5. View Request

Trigger: Submission of a View Request by a Party		
Validations performed at the:	**In case of Success**	**In case of Failure**
External Interaction Layer	View Response	HTTP error or SOAP Fault
Pre-Processing Layer	N/A	N/A
Back-office Processing	N/A	N/A

The essential parameters for the system to perform the View Request and return a human readable attached to a business document are provided below. They are all found in the message.

Element	**Location**
ID (of the document)	*DocumentReferenceRequest.ID*
Document Type Code	*DocumentReferenceRequest.DocumentTypeCode*
Sender Party Endpoint ID	*SenderParty.EndpointID*
Receiver Party Endpoint ID	*ReceiverParty.EndpointID*

The View Request must also respect the following Business Rules. If one of these rules is not respected, a SOAP Fault is returned synchronously instead of the Retrieve Response.

Business Rule	Severity
The Sender Party Endpoint ID must be present and not empty in the message payload.	*Hard*
The Receiver Party Endpoint ID must be present and not empty in the message payload.	*Hard*
The Sender Party ID present in the SOAP Business header must be equal to the Sender Party Endpoint ID or the Receiver Party Endpoint ID present in the message payload. (In other words, this rule means that the Party for which the Retrieve Request is performed is either the Sender or the Receiver of the message.)	*Hard*
Document type code is mandatory (must be present and not empty).	*Hard*
Document id is mandatory (must be present and not empty).	*Hard*

The system queries for one document matching the request criteria.

The View Response message is built based on the human readable binary attached to this document:

- when the MIME type of the human readable is **text/html** in the system, the document is returned in **HTML format** in the response
- when the MIME type of the human readable is **application/pdf** in the system, the document is returned in base **64 encoded PDF format** in the response

4.3.4.1.6. Status Request

Trigger: Submission of a Status Request by a Party		
Validations performed at the:	**In case of Success**	**In case of Failure**
External Interaction Layer	Status Response	HTTP error or SOAP Fault
Pre-Processing Layer	N/A	N/A
Back-office Processing	N/A	N/A

The essential parameters for the system to perform the Status Request of a business document are provided below. They are all found in the message.

Element	Description
Sender Party Endpoint ID	*SenderParty.EndpointID*
Receiver Party Endpoint ID	*ReceiverParty.EndpointID*
ID (of the document)	*DocumentReferenceRequest.ID*
Document Type Code	*DocumentReferenceRequest.DocumentTypeCode*

The Status Request must also respect the following Business Rules. If one of these rules is not respected, a SOAP Fault is returned synchronously instead of the Status Response.

Business Rule	Severity
The Sender Party Endpoint ID must be present and not empty in the message payload.	*Hard*
The Receiver Party Endpoint ID must be present and not empty in the message payload.	*Hard*

The Sender Party ID present in the SOAP Business header must be equal to the Sender Party Endpoint ID or the Receiver Party Endpoint ID present in the message payload. (In other words, this rule means that the Party for which the Status Request is performed is either the Sender or the Receiver of the message.)	*Hard*
Document type code is mandatory.	*Hard*
Document id is mandatory.	*Hard*

The system queries for one document matching the request criteria.

The system also queries for child documents related to that document (e.g. a Bundle document linked to its parent Bundle), excluding the document wrappers which can be retrieved as described in section 4.3.4.1.1 Retrieve Document Wrapper. The documents must pertain to a transaction set granted to the Issuer by an eventual party agreement with the Sender.

Based on the result of the query, the system builds the Status Response. For the document matching the criteria, the following information is returned in the Status Response.

Element	Description
Response Code (applicable for a response message only)	*DocumentReferenceResponseRelatedDocs.DocumentReferenceResponse.Response.ResponseCode*
Document Status Code	*DocumentReferenceResponseRelatedDocs.DocumentReferenceResponse.Status.StatusReasonCode*
ID (of the document)	*DocumentReferenceResponseRelatedDocs.DocumentReferenceResponse.DocumentReference.ID*
Issue Date (of the document)	*DocumentReferenceResponseRelatedDocs.DocumentReferenceResponse.DocumentReference.IssueDate*
Document Type Code	*DocumentReferenceResponseRelatedDocs.DocumentReferenceResponse.DocumentReference.DocumentTypeCode*
Sender Party Endpoint ID	**Identification value:** *DocumentReferenceResponseRelatedDocs.DocumentReferenceResponse.SenderParty.EndpointID* **Identification scheme Id:** *DocumentReferenceResponseRelatedDocs.DocumentReferenceResponse.SenderParty.EndpointID/@schemeID*
Sender Party Identification	**For each party identifier of the sender party:** **- the identifier value** *DocumentReferenceResponseRelatedDocs.DocumentReferenceResponse.SenderParty.PartyIdentification[i].ID* **- the identifier scheme id** *DocumentReferenceResponseRelatedDocs.DocumentReferenceResponse.SenderParty.PartyIdentification[i].ID@schemeID*
Receiver Party Endpoint ID	**Identification value:** *DocumentReferenceResponseRelatedDocs.DocumentReferenceResp*

	onse.ReceiverParty.EndpointID **Identification scheme Id:** *DocumentReferenceResponseRelatedDocs.DocumentReferenceResponse.ReceiverParty.EndpointID/@schemeID*
Receiver Party Identification	**For each party identifier of the receiver party:** **- the identifier value** *DocumentReferenceResponseRelatedDocs.DocumentReferenceResponse.ReceiverParty.PartyIdentification[i]. ID* **- the identifier scheme id** *DocumentReferenceResponseRelatedDocs.DocumentReferenceResponse.ReceiverParty.PartyIdentification[i].ID@schemeID*

When information exists about its related documents, then this information is returned for correlation purposes:

Element	Description
Response Code (applicable for a response message only)	*DocumentReferenceResponseRelatedDocs.RelatedDocument[i].Response.ResponseCode*
Document Status Code	*DocumentReferenceResponseRelatedDocs.RelatedDocument[i].Status.StatusReasonCode*
ID	*DocumentReferenceResponseRelatedDocs.RelatedDocument[i].DocumentReference.ID*
Issue Date (of the document)	*DocumentReferenceResponseRelatedDocs.RelatedDocument[i].DocumentReference.IssueDate*
Document Type Code	*DocumentReferenceResponseRelatedDocs.RelatedDocument[i].DocumentReference.DocumentTypeCode*
Sender Party Endpoint ID	**Identification value:** *DocumentReferenceResponseRelatedDocs.RelatedDocument[i].SenderParty.EndpointID* **Identification scheme Id:** *DocumentReferenceResponseRelatedDocs.RelatedDocument[i].SenderParty.EndpointID@schemeID*
Sender Party Identification	**For each party identifier of the sender party:** **- the identifier value** *DocumentReferenceResponseRelatedDocs.RelatedDocument[i].SenderParty.PartyIdentification[j].ID* **- the identifier scheme id** *DocumentReferenceResponseRelatedDocs.RelatedDocument[i].SenderParty.PartyIdentification[j]/cbc:ID/@schemeID*
Receiver Party Endpoint ID	**Identification value:** *DocumentReferenceResponseRelatedDocs.RelatedDocument[i].ReceiverParty.EndpointID* **Identification scheme Id:** *DocumentReferenceResponseRelatedDocs.RelatedDocument[i].ReceiverParty.EndpointID@schemeID*
Receiver Party Identification	**For each party identifier of the receiver party:** **- the identifier value** *DocumentReferenceResponseRelatedDocs.RelatedDocument[i].Recei*

	verParty.PartyIdentification[j].ID **- the identifier scheme id** *DocumentReferenceResponseRelatedDocs.RelatedDocument[i].Receiverparty.PartyIdentification[j].ID@schemeID*

4.3.4.1.7. Retrieve Interchange Agreements Request version 2.0

Trigger: Submission of a Retrieve Interchange Agreement Request by a Party		
Validations performed at the:	**In case of Success**	**In case of Failure**
External Interaction Layer	Retrieve Interchange Agreements Response	HTTP error or SOAP Fault
Pre-Processing Layer	N/A	N/A
Back-office Processing	N/A	N/A

The essential parameters for the system to perform the Retrieve Interchange Agreements Request are provided below and they are all found in the message:

Element	**Location**
Request Sender Party Identifier	*Header.BusinessHeader.Sender.Identifier*
Sender Party Endpoint ID	*SenderParty.EndpointID*
Receiver Party Endpoint ID	*ReceiverParty.EndpointID*
Document Type Code	*DocumentTypeCode*

The Retrieve Interchange Agreements Request must also respect the following Business Rules. If one of these rules is not respected, a SOAP Fault is returned synchronously instead of the Retrieve Interchange Agreements Response.

Business Rule	**Severity**
The number of document type codes specified in the request must not be greater than 500.	*Hard*

The system queries for the interchange agreements with a specific sender and receiver and concerning documents with the type codes specified where either the sender or the receiver is the same with the request sender party.

Based on the result of the query, the system builds the Retrieve Interchange Agreements Response. For the interchange agreements matching the criteria, the following data is returned:

Element	**Location**
Sender Party Endpoint ID	*InterchangeAgreement.SenderParty.EndpointID*
Set of Sender Party Identifiers with their scheme ID	*InterchangeAgreement.SenderParty.PartyIdentification.ID*
Receiver Party Endpoint ID	*InterchangeAgreement.ReceiverParty.EndpointID*
Set of Receiver Party Identifiers with their scheme ID	*InterchangeAgreementReceiverParty.PartyIdentification.ID*
Confidentiality Level	*InterchangeAgreement.SecurityInformation.ConfidentialityLevelCode*
Integrity Level	*InterchangeAgreement.SecurityInformation.IntegrityLevelCode*

Availability Level	*InterchangeAgreement.SecurityInformation.AvailabilityLevel Code*
Sender Party Certificate	*N/A*
Receiver Party Certificate	*InterchangeAgreement.SecurityInformation.ReceiverPartyCertificate* – a list of certificates depending on the key usage. Currently, the system maps only certificates having key usage KEY_ENCIPHERMENT and will choose the latest certificate which is active and not revoked.
Profile Name	*InterchangeAgreement.ProfileID*

The possible values of the elements Confidentiality Level, Integrity Level, Availability Level, Party Certificate Key Usage and Profile Name can be found in [REF10].

The response message is digitally signed by the e-TrustEx before being sent to the request sender party.

4.3.4.1.8. Retrieve Interchange Agreements Request version 2.1

Trigger: Submission of a Retrieve Interchange Agreement Request by a Party		
Validations performed at the:	**In case of Success**	**In case of Failure**
External Interaction Layer	Retrieve Interchange Agreements Response	HTTP error or SOAP Fault
Pre-Processing Layer	N/A	N/A
Back-office Processing	N/A	N/A

The essential parameters for the system to perform the Retrieve Interchange Agreements Request are provided below and they are all found in the message:

Element	**Location**
Request Sender Party Identifier	*Header.BusinessHeader.Sender.Identifier*
Sender & Receiver Party ID and scheme	*Party[i].PartyIdentification.ID and* *Party[i].PartyIdentification.ID@schemeID*
Document Type Code	*DocumentTypeCode*

The Retrieve Interchange Agreements Request must also respect the following Business Rules. If one of these rules is not respected, a SOAP Fault is returned synchronously instead of the Retrieve Interchange Agreements Response.

Business Rule	**Severity**
The number of document type codes specified in the request must not be greater than 500.	*Hard*
Maximum one Party Identifier per party element is allowed.	*Hard*
If the party identifier schemeId attribute is not present or contains an empty value, GLN schemeId shall be used.	*Hard*
When a Party Identifier is specified, it shall have a corresponding party in the System.	Hard

The system queries for the interchange agreements involving the specified parties and concerning documents with the type codes specified where either the sender or the receiver is the same with the request sender party.

Based on the result of the query, the system builds the Retrieve Interchange Agreements Response. For the interchange agreements matching the criteria, the following data is returned:

Element	Location
Party Identifiers and their type for each party involved	**Identifier:** *InterchangeAgreement[i].PartyRole[j].Party.PartyIdentification.ID* **Type:** *InterchangeAgreement[i].PartyRole[j].Party.PartyIdentification.ID@schemeID*
Party Names for each party involved	*InterchangeAgreement[i].PartyRole[j].Party.PartyName.Name*
Party Role Code for each party involved	*InterchangeAgreement[i].PartyRole[j].RoleCode*
Confidentiality Level	*InterchangeAgreement[i].SecurityInformation.ConfidentialityLevelCode*
Integrity Level	*InterchangeAgreement[i].SecurityInformation.IntegrityLevelCode*
Availability Level	*InterchangeAgreement[i].SecurityInformation.AvailabilityLevelCode*
Profile name	*InterchangeAgreement[i].ProfileID*
Party Certificate of each party involved	*InterchangeAgreement[i]. PartyRole[j].Party.X509Certificate* Will hold the certificate having key usage KEY_ENCIPHERMENT, if one exists, is active and not revoked.
Validity start date	*InterchangeAgreement[i].ValidityPeriod.StartDate*
Validity start time	*InterchangeAgreement[i].ValidityPeriod.StartPeriod*

The possible values of the elements Confidentiality Level, Integrity Level, Availability Level, Party Certificate Key Usage and Profile Name can be found in [REF10].

The response message is digitally signed by the e-TrustEx before being sent to the request sender party.

4.3.4.2. Operational Services

4.3.4.2.1. Store Document Wrapper

Trigger: Submission of a Store Document Wrapper Request by a Party		
Validations performed at the:	**In case of Success**	**In case of Failure**
External Interaction Layer	Technical Ack	HTTP error or SOAP Fault
Pre-Processing Layer	N/A	N/A
Back-office Processing	N/A	N/A

The processing of a Store Document Wrapper request is fully synchronous and should be considered as a single, atomic step. At the end of the processing the binary file representing the document wrapper and the message are saved in the System's database. The SOAP header is also saved as a binary attached to the message. The message

will have the type provided by the Sender in the message payload and will be referencing its corresponding binary file.

The Store Document Wrapper Request must respect the following Business Rules. If one of these rules is not respected, a SOAP Fault is returned synchronously instead of the saving of the message and binary.

Business Rule	**Severity**
The DocumentWrapper ID must not be empty	*Hard*
The DocumentWrapper ID must not be longer than 250 characters	*Hard*
The DocumentWrapper ID must only have ASCII characters	*Hard*
Another message with the same DocumentWrapper ID and type must not have been sent previously by the same Sender Party.	*Hard*
The size specified in the DocumentSize element of the request is the same with the size of the binary file.	*Hard*
The size specified in the attribute request is the same with the size of the binary file.	*Hard*
The size of the binary file must not exceed the maximum size configured (SLA policy).	*Hard*
By storing the binary file, the maximum volume allowed must not be exceeded (SLA policy).	*Hard*

4.3.4.2.2. Delete Document Wrapper

Trigger: Submission of a Delete Document Wrapper Request by a Party		
Validations performed at the:	**In case of Success**	**In case of Failure**
External Interaction Layer	Technical Ack	HTTP error or SOAP Fault
Pre-Processing Layer	N/A	N/A
Back-office Processing	N/A	N/A

The processing of a Delete Document Wrapper request is fully synchronous and should be considered as a single, atomic step. At the end of the processing the binary file representing the document wrapper and the Document Wrapper message corresponding to it are deleted from the System's database.

The Delete Document Wrapper Request must respect the following Business Rules. If one of these rules is not respected, a SOAP Fault is returned synchronously instead of the deleting the corresponding Document Wrapper message and its binary.

Business Rule	**Severity**
The DocumentWrapper ID must not be empty	*Hard*
The DocumentWrapper ID must not be longer than 250 characters	*Hard*
The DocumentWrapper ID must only have ASCII characters	*Hard*
The DocumentWrapper DocumentTypeCode is not empty	*Hard*
A Document Wrapper message with the same DocumentWrapper ID and type, and sent by the same Sender Party must already exist.	*Hard*
The Document Wrapper message must not be linked to a Document Bundle.	*Hard*

4.3.4.2.3. Delete Document

Trigger: Submission of a Delete Document Request by a Party		
Validations performed at the:	**In case of Success**	**In case of Failure**
External Interaction Layer	Technical Ack	HTTP error or SOAP Fault
Pre-Processing Layer	N/A	N/A
Back-office Processing	N/A	N/A

The processing of a Delete Document request is fully synchronous and should be considered as a single, atomic step. At the end of the processing the document and all its children that have no other parent than the document identified in the request will be deleted from the System's database. The binaries attached to these documents will also be deleted.

An event notification for the document deletion may be sent to the concerned parties, as described in section 4.3.3.4 Event Notification.

The Delete Document Request must respect the following Business Rules. If one of these rules is not respected, a SOAP Fault is returned synchronously instead of the deleting the corresponding Document message and its children.

Business Rule	**Severity**
The Document Sender Party Identifier (DocumentSenderParty/EndpointID) must be present and not empty.	*Hard*
The Document Receiver Party Identifier (DocumentReceiverParty/EndpointID) must be present and not empty.	*Hard*
The Sender Party (identified from the business header information) is allowed to perform document deletion.	*Hard*
The Sender Party Identifier present in the SOAP Business header must be equal to the Sender Party ID or the Receiver Party ID present in the message payload.	*Hard*
Document Id (DocumentReference.ID) must be present and not empty.	*Hard*
Document Type Code (DocumentReference.DocumentTypeCode) must be present and not empty.	*Hard*

4.3.4.2.4. Create Party

Trigger: Submission of a Create Party Request by a Party		
Validations performed at the:	**In case of Success**	**In case of Failure**
External Interaction Layer	Technical Ack	HTTP error or SOAP Fault
Pre-Processing Layer	N/A	N/A
Back-office Processing	N/A	N/A

The processing of a Create Party request is fully synchronous and should be considered as a single, atomic step. At the end of the processing the party, its associated party identifiers, party certificate and credentials are created in the System's database.

An event notification for the Party creation may be sent to the concerned parties, as described in section 4.3.3.4 Event Notification.

The Create Party Request must respect the following Business Rules. If one of these rules is not respected, a SOAP Fault is returned synchronously instead of the creating the Party and its corresponding party identifiers.

Business Rule	Severity
Party Identifier: The Party must have at least one party identification element. It can be found via the following xPath: *Party/PartyIdentification[i]*.	*Hard*
Party Identifier ID: The Party Identification ID element must contain a value. It can be found via the following xPath: *Party/PartyIdentification[i]/ID.*	*Hard*
Party Identifier schemeID: When present, the value of the schemeID attribute must be part of a list of known identifier types. It can be found via the following xPath: *Party/PartyIdentification[i]/ID/@schemeID*	*Hard*
There must be one and only one party name element. It can be found following the xPath: *Party/PartyName.*	*Hard*
The PartyName Name element must contain a value. It can be found via the following xPath: *Party/PartyName/Name*	*Hard*
The values found in the following elements must be at most 255 characters long: • *Party/PartyIdentification[i]/ID* • *Party/PartyIdentification[i]/ID/@schemeID* • *Party/PartyName/Name* • *Party/PartyCredentials/Username* • *Party/PartyCredentials/Password*	*Hard*
When Party Credentials Password element is present its encrypted attribute must contain a value. It can be found via the following xPath: *Party/PartyCredentials/Password/@encrypted.*	*Hard*
The value found in the PartyName Name element is unique in the business domain of the message issuer (xPath: *Party/PartyName/Name).*	*Hard*
Party Identifier scheme id: There cannot be two or more party identifiers with the same schemeID value. If no schemeID, GLN will be considered by default.	*Hard*
Party Identifier uniqueness: The party identifier (value and scheme Id) must be unique per business domain. E.g.: there shall be no two parties pertaining to same business domain and having the same VAT number.	*Hard*
If ThirdPartyIndicator element is present and its value is "true", then PartyCredentials element is mandatory. It can be found following the xPath: *Party/PartyCredentials*	*Hard*
When present, the party credentials username element must contain a value. It can be found via the following xPath: *Party/PartyCredentials/Username*	*Hard*
Party Credentials Username Uniqueness The Credentials username must be unique amongst usernames of party	

Credentials.	
When present, the party credentials password element must contain a value. It can be found via the following xPath: *Party/PartyCredentials/Password*	*Hard*
If a password is present, it must be Base64 encoded.	*Hard*
The size of the party certificate must not exceed 100k. The certificate can be found via the following xPath: *Party/X509Certificate*.	*Hard*

4.3.4.2.5. Create Interchange Agreement

Trigger: Submission of a Create Interchange Agreement Request by a Party		
Validations performed at the:	**In case of Success**	**In case of Failure**
External Interaction Layer	Technical Ack	HTTP error or SOAP Fault
Pre-Processing Layer	N/A	N/A
Back-office Processing	N/A	N/A

The processing of an Interchange Agreement request is fully synchronous and should be considered as a single, atomic step. At the end of the processing the Interchange Agreement and its associated entities (e.g. party role associations, party agreements) are created in the System's database.

An event notification for the Interchange Agreement creation may be sent to the concerned parties, as described in section 4.3.3.4 Event Notification.

The Create Interchange Agreement Request must respect the following Business Rules. If one of these rules is not respected, a SOAP Fault is returned synchronously instead of the creating the Interchange Agreements and its associated entities.

Business Rule	**Severity**
The Profile element must not contain an empty value. It can be found via the following xPath: *InterchangeAgreement/ProfileID*	*Hard*
The value in the Profile element must have an accepted value. *	*Hard*
Exact two PartyRole elements must be specified. PartyRole elements can be found via the following xPath *InterchangeAgreement/PartyRole[i]*.	*Hard*
For each PartyRole element, the RoleCode must not be empty. It can be found via the following xPath: *InterchangeAgreement/PartyRole[i]/RoleCode*	*Hard*
For each PartyRole element, the RoleCode must have an accepted value.*	*Hard*
Each Party or Third Party (if one is specified) must have one and only one PartyIdentification element. It can be found via the following xPath: • *InterchangeAgreement/PartyRole[i]/Party/PartyIdentification[j]* for the parties in the interchange agreement • *InterchangeAgreement/PartyRole[i]/Party/AgentParty/PartyIdentification[j]* for the third parties	*Hard*
For each Party or Third Party (if one is specified), the PartyIdentification ID element must not be empty. It can be found via the following xPath: • *InterchangeAgreement/PartyRole[i]/Party/PartyIdentification[j]/ID* for the parties in the interchange agreement • *InterchangeAgreement/PartyRole[i]/Party/AgentParty/PartyIdentific*	*Hard*

ation[j]/ID for the third parties	
If present, the validity start date must not be in the past. It can be found via the following xPath: *InterchangeAgreement/ValidityPeriod/StartDate*	*Hard*
The Confidentiality Level Code value must have an accepted value.* It can be found via the following xPath: *InterchangeAgreement/SecurityInformation/ConfidentialityLevelCode*	*Hard*
The Integrity Level Code value must have an accepted value.* It can be found via the following xPath: *InterchangeAgreement/SecurityInformation/IntegrityLevelCode*	*Hard*
The Availability Level Code value must have an accepted value.* It can be found via the following xPath: *InterchangeAgreement/SecurityInformation/AvailabilityLevelCode*	*Hard*
A Profile with the same name must already be configured within the platform and associated with the Issuer Party business domain.	*Hard*
There must be one and only one party identifier configured in the system having the type equal to the schemeID attribute and the value equal to the one specified in the PartyIdentification element. If the schemeId value is not present or has an empty value, "GLN" will be used as default.	*Hard*
The party corresponding to each PartyIdentification element must belong to the Issuer Party business domain and must not be flagged as a third party.	*Hard*
If mentioned, any third party (identified by an agent PartyIdentification element) shall pertain to the Issuer Party business domain and be flagged as third party in the System.	*Hard*
If the RoleCode is the same for more than one PartyRole element then the role corresponding to it must be bidirectional in the System.	*Hard*
The combination of roles matches a combination of roles in the profile's transactions.	*Hard*
If the following conditions are not met the interchange agreement shall not be created, as the party will not be able to connect in order to send messages: • a party does not have credentials in the System **and** • no third party is specified in the request **and** • the party has no party agreement with any third party configured in the System	*Hard*
No other interchange agreement having • The same profile **and** • The same parties in exactly the same roles	*Hard*

shall exist in the System. This is a duplicate check.	

*The possible values of elements Profile, Role Code, Confidentiality Level Code, Integrity Level Code and Availability Level Code can be found in the tabs ProfileCode, RoleCode, ConfidentialityLevelCode, IntegrityLevelCode respectively AvailabilityLevelCode in e-TrustEx Code List [REF10].

4.3.5. Service versioning

There are countless reasons why Services need to evolve after being live. A few of them include:

- Bugs may need to be fixed.
- Changing requirements.
- Different flavours of a Service may be desirable.

Whatever the cause, over time, the service will evolve and this will mandate the versioning of the interface's XML Documents beyond construction.

The most important aspects of the evolution are captured in the table below:

Table 4 Service versioning

Type of Evolution	Re-use of the document's namespace names?	Required migration of old versions?
Backwards compatible evolution	Yes	No
Backwards semi-compatible evolution	Yes	Yes
Backwards incompatible evolution	No	Yes

4.3.5.1. Backward compatible evolution (most likely for Read Services)

An evolution is backward compatible when the newer version of a service interface can be rolled-out without breaking its existing consumers. Examples of such changes are provided in the table below:

Table 5 Backward Compatible Change

Artefact	Backward Compatible Change
XSD	• Adding optional elements or attributes to a namespace • Increasing the maximum allowed number of occurrences of an element or attribute.
Schematron	• Adding business rules which only produce warnings. • Extending the content of a Code Table.
WSDL	• Adding a SOAP Header Block which can be ignored.

This means that the service's interface is extended in a way that its previous versions continue to be supported.

When the new version of the service supports its usage by old and new versions of its consumers, the optimal evolution strategy is simply to **update** the service when required. This scenario is most likely to be case for the evolution of the Read Services.

In this case the re-use of the document's namespace names is possible. If a backward compatible change is made to a service exposed by e-TrustEx, then the old namespace name is used.

4.3.5.2. Semi-backward compatible evolution (most likely for Write Services)

An evolution is semi-backward compatible when the newer version of a service imposes additional mandatory constraints to existing consumers.

Artefact	Incompatible Change

Business Rules	• Restricting the content model via a hard constraint. • Restricting the content of a Code Table.

In this case the re-use of the documents' namespace names is possible. If a semi-backward compatible change is made to a service exposed by e-TrustEx, then the old namespace name is used.

4.3.5.3. Backward incompatible evolution (most likely for Write Services)

An evolution is incompatible when the newer version of a service interface imposes additional mandatory constraints to existing consumers. Examples of such changes are provided in the table below:

Table 6 Incompatible evolution

Artefact	Incompatible Change
XSD	• Adding of mandatory elements to an XML Schema. • Restricting the XSDs content model, such as changing a choice to a sequence in XML. • Removing mandatory elements to an XML Schema. • Significant semantic changes to existing core components.
WSDL	• Adding a SOAP Header Block which must be understood.

When the new version of the service impacts the interpretation and validation of the old interface's XML Documents, the service can no longer be used by old consumers. In this case multiple versions of the service will be kept running in the same runtime environment until the old one is made unavailable. In this case two (or more) versions of the same service coexist.

In this case the XML parser linked to the new version of the service will require that the new version and related schemas are used. Therefore, users will be required to adapt their implementation of the interface before the service is made unavailable.