
# Dedicated Links Connectivity Specifications

**V0.1**

| | |
|---|---|
| Author | 4CB |
| Version | V0.1 |
| Date | 28/10/2011 |

# 1. Introduction

To be completed later.

# 2. The network connectivity

To be completed later.

# 3. The MQ configuration

To be completed later.

# 4. Data Exchange Protocol (DEP) and messaging services

T2S connectivity model has to interface several counterparts (VA Network Service Providers and DCPs).

In order to manage the message communication with a multiplicity of actors, T2S adopts an ad hoc protocol for data exchange in A2A mode. The Data Exchange Protocol (DEP) is used to exchange data between the T2S platform and the NSP-VA or the DCP connected via a NSP-DL channel. This protocol is developed on the MQ application-to-application protocol, so to inherit some of the native functionalities provided by the MQ product (as explained in section 1.1.1.1.3 MQMD message format) .

The Data Exchange Protocol ensures independence from vendors and interoperability also for the future. Indeed, the aim of this protocol is to let the T2S network interface being independent from the network protocol used by the NSP or by the DCP.

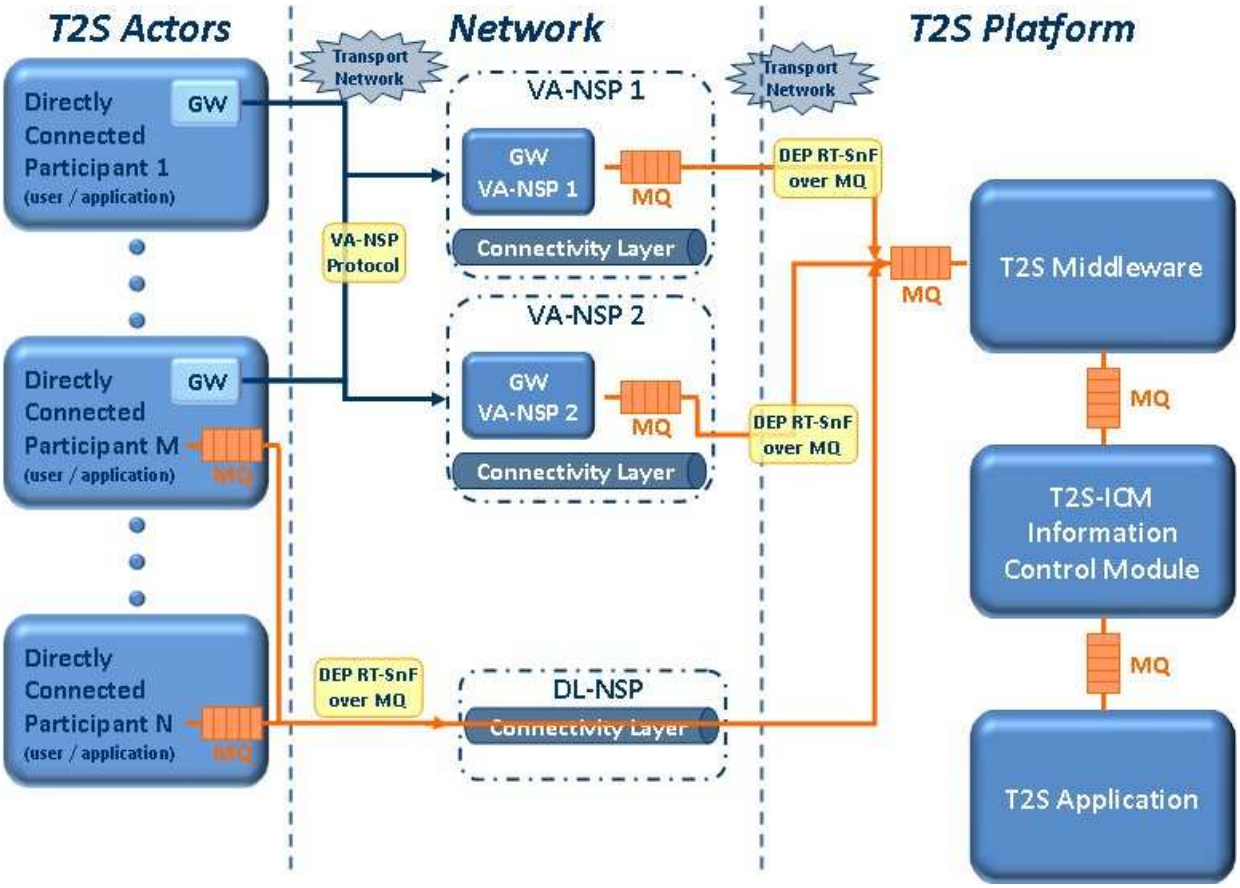
## 4.1. Communication schema and data exchange scenario

DIAGRAM 1 below shows the overall flow for the inbound and outbound data exchanged between DCPs and the T2S Platform.

DCPs can be classified into three categories:

1. DCPs which communicate with the T2S platform through a VA-NSP intermediation (e.g. Actor 1 in the diagram below). In this case, the data exchange is compliant with a protocol defined by the relevant NSP and it is managed by the gateway of the DCP (i.e. the original sender) and the gateway of the NSP. Then, the VA-NSP offers connectivity services and manages the bi-directional data exchange with T2S Platform according to the DEP, using real-time or store-and-forward data exchange. DEP messages are encapsulated in MQ standard messages. The VA-NSP offers several functionalities: authentication, CGU, authorization, non-repudiation, transformation and DEP management support.
2. DCPs which communicate directly with T2S platform, using a DL-NSP only for transport connectivity services (e.g. Actor N in the diagram). In this case, the DCP and the T2S platform communicate directly according to the DEP, using real-time or store-and-forward data exchange. DEP messages are encapsulated in MQ standard messages.
3. DCPs which can communicate both via the intermediation of a VA-NSP (category 1) or directly via a DL-NSP (category 2). In this case, the DCP (e.g. Actor M in the diagram) has to define a gateway for VA-NSP communication, in compliance with communication protocol adopted by the relevant NSP, plus an MQ interface for DEP data exchanges over the MQ connection.

**DIAGRAM 1: DATA EXCHANGE BETWEEN DCPS AND THE T2S PLATFORM**




The T2S platform's logical architecture comprises a first level, named T2S middleware component, which represents the interface towards DCPs. At T2S platform level, the data exchange with external users and applications is performed exclusively via DEP messages over MQ channels.

For the exchange of inbound data sent by DCPs/VA-NSPs, the T2S middleware component is responsible for envelope validation, non-repudiation and technical ack management, decompression of the data and building the relevant input parameter list for the Interface to application component (ICM[1]).

The T2S middleware component acts as a boundary towards the T2S application layer. When the T2S application layer receives the business payload and the input parameter list from the T2S middleware, this layer is responsible for several functionalities: validation, authorization, transformation, non repudiation management, de-bulking and routing to the T2S application module.

Finally, the T2S application receives and processes business data.

[1] Information and Control Module.

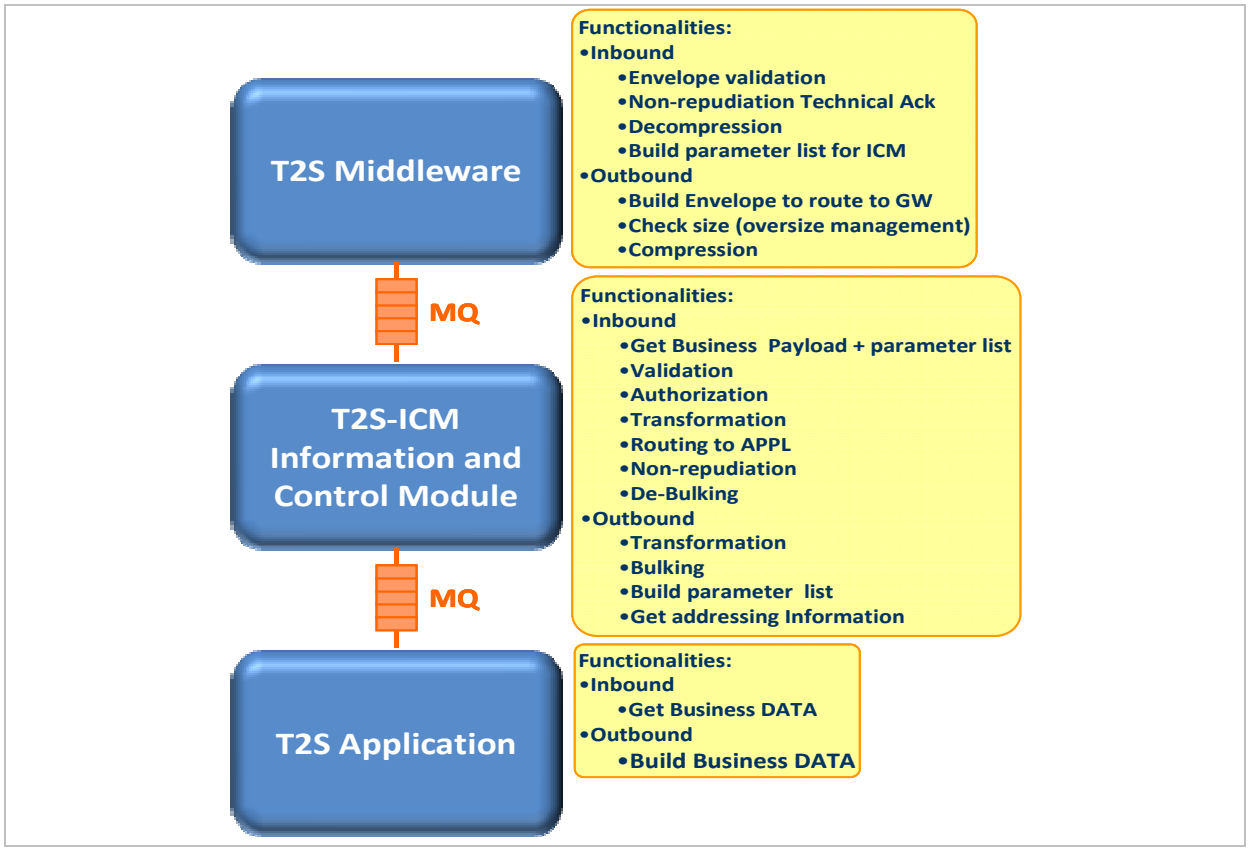
The T2S application is the sender of the original request in the outbound data exchange scenario, building the business data that the T2S Platform sends to the relevant DCP (optionally via the intermediation of a VA-NSP).

Then, the T2S application component sends business data to the ICM component. In this scenario, ICM is responsible to get and to manage several processing functionalities: transformation, bulking, building the output parameter list.

After that, data are exchanged between ICM and the T2S middleware component. The T2S middleware compresses the data to be sent to the DCP, checking the final size of the message/file by means of oversized data management functionalities (see section 1.7.2 – Oversize Data Management). As a final step, the T2S middleware builds the envelope and routes it to the NSP gateway (in a VA-NSP scenario) or to the DCP MQ Interface (in a DL-NSP scenario).

The main T2S components and inbound/outbound functionalities are depicted in the following diagram.

**DIAGRAM 2 - T2S PLATFORM FUNCTIONALITIES OVERVIEW**



## 4.2. MQMD message format

VA-NSPs and DCPs directly connected with T2S platform shall connect to the WebSphere MQ (WMQ) architecture of the T2S platform in order to manage services available in A2A mode. For each VA-NSP/DCP, the T2S platform provides at least one WMQ queue  each of the following data flow types:

- real-time messages;
- real-time files;
- store-and-forward messages;
- store-and-forward files.

The T2S middleware and the VA-NSP/DCP manage the data exchange based on WMQ messages, that consist a Message Description part (MQMD) and a Message Text part.

As previously mentioned, the message structure of DEP uses standard MQ messages; so, DEP uses IBM WebSphere MQ as transport layer for its message primitives. The message text is in XML.

**DIAGRAM 3: MQ DATA STRUCTURE**

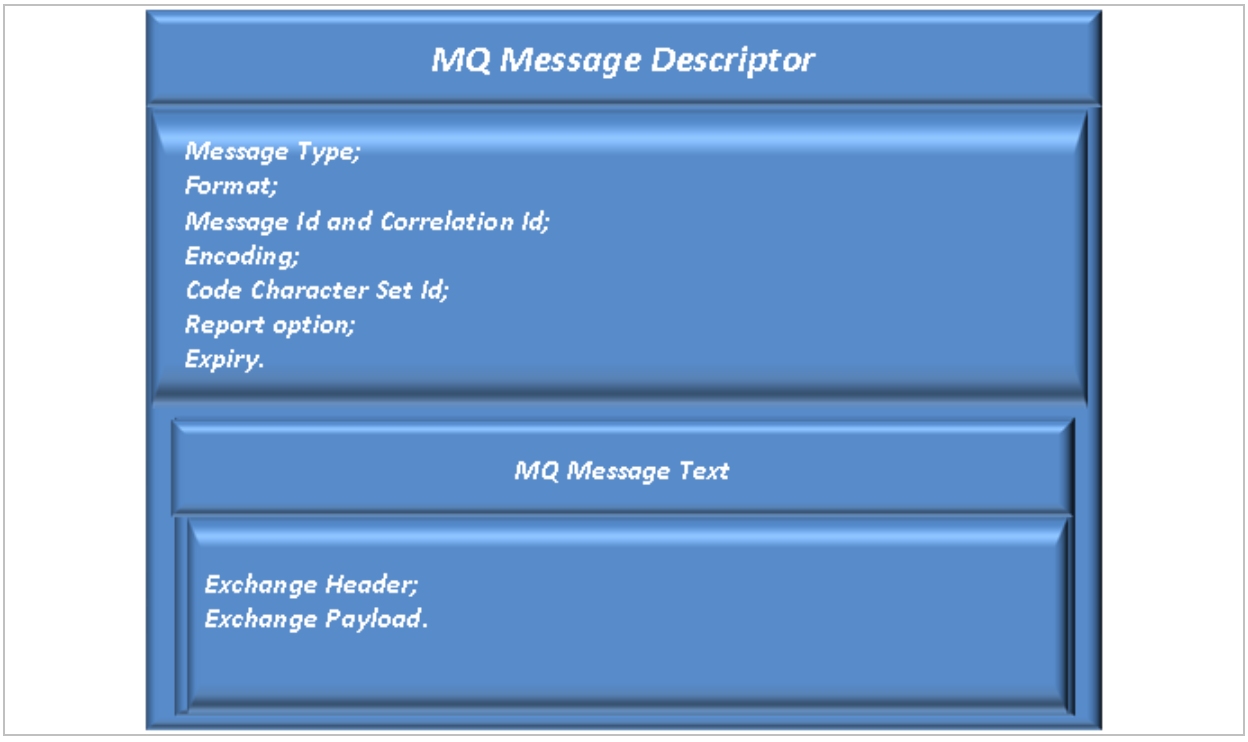



Diagram 3 shows the MQ format message structure:

- MQ message descriptor: it contains all the data required to manage the message from an infrastructure point of view.

- MQ message text: it is the information content of the message, formed according to the DEP structure (detailed in the following section), i.e.:
  - DEP header (the Exchange Header of the DEP data structure);
  - Business message (composed by the Business Application Header and by the business payload);
  - The digital signature of the sender (T2S Platform or DCP/NSP) of the message, based on the "DEP Exchange Header + Business Message" content. This structure is present only if the "Non Repudiation" flag is set in the exchange header of the DEP data exchange.

TABLE 1 describes the list of WMQ message standard MQMD header fields, which the NSP/DCP and the T2S platform manage when a message or a file is exchanged, and the contents of the "Message Text" part, in order to support the DEP message primitives.

**TABLE 1: MQ DATA STRUCTURE**

| DATA STRUCTURE | DESCRIPTION |
|---|---|
| **MQMD section** | No particular header (e.g. RFH2) are foreseen.<br>• **MQMD.MsgType**: request/reply/report/datagram values are allowed;<br>• **MQMD.Format**: e.g. MQFMT_NONE;<br>• **MQMD.MsgId** and **CorrelationId**;<br>• **MQMD.Encoding**;<br>• **MQMD.CodeCharacterSetId**;<br>• **MQMD.Report** option: set to the value MQRO_PAN+MQRO_NAN;<br>• **MQMD.Expiry**: this field could be used only for real-time traffic setting the value equal to the real-time time-out timeframe (e.g. 60 seconds). In this way it is possible to avoid unnecessary management of messages already expired. |
| **MQ Message Text** | • **Exchange Header** section: contains all "service information" needed for the transport layer, exchanged between the DCP and the T2S Platform to manage messages and files flows;<br>• **Exchange Payload** for business layer: contains the Business Envelope with document (or document set) section. |

The DEP requires a technical ack message, i.e. an acknowledgement provided for each data exchange between the T2S platform and the DCP for confirmation of the completion of the data exchange.

This technical ack is a WebSphere MQ report message with type PAN (Positive Application Notification) or NAN (Negative Application Notification). The receiving counterpart, i.e the T2S middleware or the DCP middleware, sends back this acknowledgment when it undertakes the received message (e.g. when it stores the message or it starts processing it).

The structure of the technical ack is as follows:

1. The MQMD.Feedback field of the MQ Message Descriptor returns the value 0 (zero) in the case a of PAN or a positive numeric value in the case of a NAN;

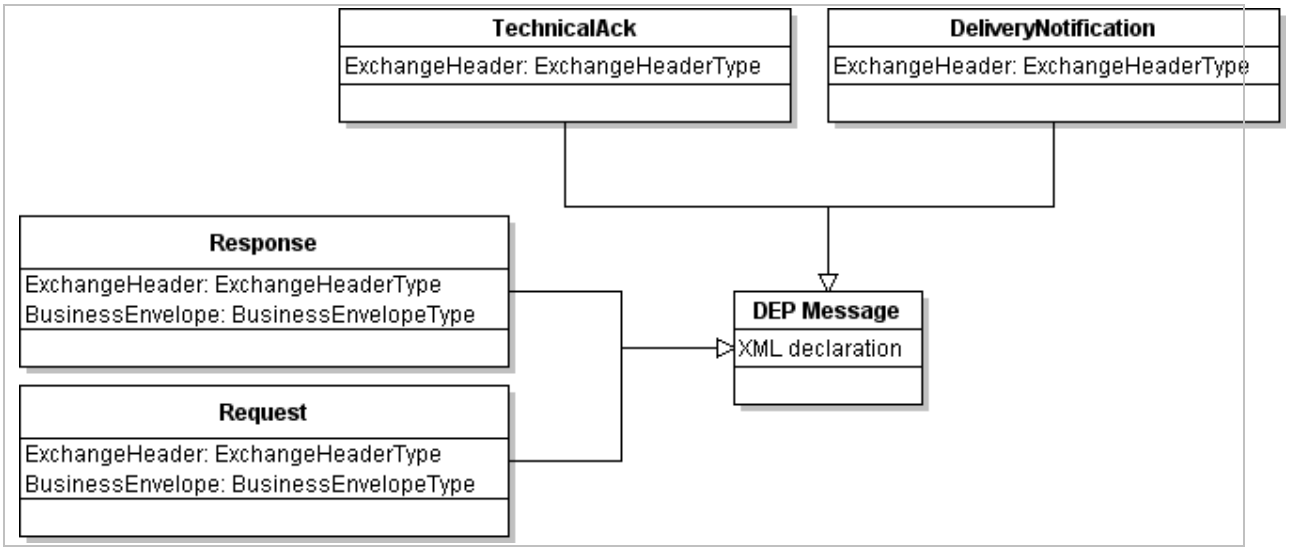
2. The Application Identity Data of the MQMD, returns the system identification of the receiving application (the NSP's/DCP's gateway hostname or the T2S platform hostname);
3. The Correlation Id field of the MQMD section returns the "Message Id" value of the original message;
4. The Message Text part of the MQ message returns the "Exchange Header" of the original message, updated as foreseen in the following message patterns description in the case of a PAN.

In the case of a NAN the field "dep:ErrorDescription" returns an error message.

## 4.3. DEP message format

A valid DEP message is a well-formed XML document, as defined in W3C specification, meeting some additional constraints expressed by an XML Schema Definition (XSD, ref. Appendix 1). Every message presents an XML declaration, which specifies the version of XML being used (*version="1.0"*), the Character Encoding (*encoding="UTF-8"*) and an XML element which defines the message primitive.

**DIAGRAM 4 : MAIN DEP MESSAGE PRIMITIVES.**




As shown in DIAGRAM 4, the DEP specifies four main message primitives, each one with a proper XML element type definition:

- A Request message: the T2S platform uses this message type to send a message/file to a DCP, and vice versa. This kind of primitive is used both in real-time mode and in store-and-forward mode.
- A Response message: the T2S platform uses this message type to answer a previously received request. This kind of primitive is used only in real-time mode.

- A Technical Ack message: this acknowledgement is provided for each data exchange between the T2S platform and a DCP for the confirmation of the completion of the data exchange. This kind of primitive shall be used both in real-time mode and in store-n-forward mode. For non repudiation, the receiver must sign the Technical Ack and include the hash of the received message and the relevant timestamp in the ExchangeHeader.
- A Delivery Notification message: in store-and-forward mode, after 10 unsuccessful attempts, the VA-NSP sends back to the original sender a "Delivery Notification Failure" and suspends the sending of the store-and-forward messages/files to the T2S platform.

Complying with these type definitions, the XML element of these DEP messages consists of two parts: a header (mandatory for every message) and a payload (required for request/response messages).

**DIAGRAM 5 - DEP EXCHANGEHEADER TYPE.**

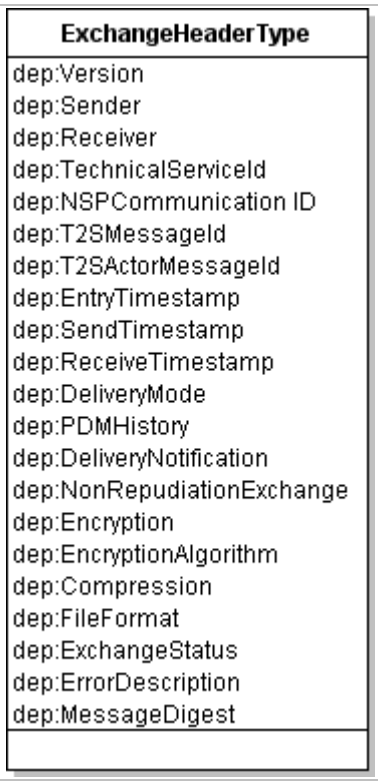



**DIAGRAM 6 - DEP BUSINESSENVELOPE TYPE (PAYLOAD).**


The Exchange Header section (see Diagram 5) contains all service information needed by the transport layer, exchanged between the counterpart (NSP-VA or DCP) and the T2S platform to manage messages and files flows.

The Exchange Payload for business layer (BusinessEnvelope + document or document set) section (see Diagram 6 ) contains business information. This part is not checked or modified by the NSP-VA or by the T2S middleware and it is delivered unchanged to the receiver.

An example of a DEP protocol message is shown hereafter:

```
<?xml version="1.0" encoding="UTF-8" ?>
<dep:Request
 xmlns:dep="http://www.ecb.eu/t2s-0.2"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.ecb.eu/t2s-0.2 dep-02.xsd ">
       <dep:ExchangeHeader>
              <dep:Version>0.2</dep:Version>
              <dep:Sender>cn=t2sappl1,o=t2sprod</dep:Sender>
              <dep:Receiver>cn=t2s-cust1,o=nsp-name1</dep:Receiver>
              <dep:TechnicalServiceId>nsp-name1.MSGRT.PROD</dep:TechnicalServiceId>
              <dep:T2SMessageId>
                     T2S.MSGRT.NSPname1.20110101000000.000001
              </dep:T2SMessageId>
              <dep:SendTimestamp>2011-01-01T00:00:00</dep:SendTimestamp>
              <dep:DeliveryMode>RT</dep:DeliveryMode>
              <dep:ExchangeStatus>OK</dep:ExchangeStatus>
       </dep:ExchangeHeader>
       <dep:BusinessEnvelope>
              <dep:BusinessApplicationHeader>
                     <!-- business application header goes here -->
              </dep:BusinessApplicationHeader>
              <dep:BusinessMessage>
                     <!-- business message goes here -->
              </dep:BusinessMessage>
       </dep:BusinessEnvelope>
</dep:Request>
```

## 4.4. DEP exchange header fields description

As shown in DIAGRAM 5 , the Exchange Header section contains all service information needed for the transport layer, exchanged between a counterpart (NSP-VA or DCP via a direct link) and the T2S platform to manage messages and files flows, in real-time or store-and-forward mode. As specified in the XSD (XML Schema Definition) of the DEP Exchange Header, provided in the appendix 1, this section contains many mandatory or optional fields, in case of DEP message type.

The DEP uses a technical header (ExchangeHeader) to convey technical information to the counterpart. This technical information is used for many purposes, i.e.:

- Addressing (dep:Sender and dep:Receiver);
- Selecting Message/File channel to use (dep:TechnicalServiceId);
- Message/File identifier (dep:T2SActorMessageId, dep:T2SMessageId);
- Timestamps (dep:SendTimestamp, dep:ReceiveTimestamp);
- Function's flags (dep:DeliveryNotification, dep:NonRepudiationExchange, dep:Encryption and dep:Compression);
- Message/File security (dep:Digest);
- Message/File flow control (dep:ExchangeStatus, dep:ErrorDescription).

TABLE 2 includes an exhaustive list of the available tags and their descriptions, specifying allowed values and providing an example for each tag.

**TABLE 2: DEP EXCHANGEHEADER FIELDS.**

| TAG NAME | TAG DESCRIPTION / ALLOWED VALUES | EXAMPLE |
|---|---|---|
| **dep:Version**<br>*<mandatory tag>* | Version of Data Exchange Protocol. **Enumeration** with fixed value "**0.2**" | <dep:Version><br>0.2<br></dep:Version> |
| **dep:Sender**<br>*<mandatory tag>* | NSP user identification for T2S System User that sends the message. **Restriction** is set on base type "**string [100]**". | <dep:Sender><br>cn=t2sappl1,<br>o=t2sprod<br></dep:Sender> |
| **OriginalSender**<br>*<minOccurs="0">* | **Restriction** on base type "**string [100]**" (only for future use). | |
| **dep:Receiver**<br>*<mandatory tag>* | NSP user identification for T2S System User that receives the message. **Restriction** is set on base type "**string [100]**". | <dep:Receiver><br>cn=t2s-cust1,<br>o=nsp-name1<br></dep:Receiver> |
| **FinalReceiver**<br>*<minOccurs="0">* | **Restriction** on base type "**string [100]**" (only for future use). | |
| **dep:TechnicalServiceID**<br>*<mandatory tag>* | Name of the service used to send messages and files, formed by the Network Service Provider name, the message pattern and the environment of reference.<br>Specifying a message pattern, it's possible to manage a message or a file as a payload of the DEP message. **Message pattern** meaning is the following:<br>• MSGRT: **Real Time Message**;<br>• MSGSNF: **Store & Forward Message**;<br>• FILERT: **Real Time File**;<br>• FILESNF: **Store & Forward File**.<br>**Restriction** is set on base type "**string [60]**", with expression in the format: **[NSP_name].[msg_pattern].[environment]**<br>where msg_pattern= {MSGRT \| MSGSNF \| FILERT \| FILESNF} and environment= {EAC \| UTEST \| PROD…}. | <dep:TechnicalServiceId><br>nsp-name1.MSGRT.PROD<br></dep:TechnicalServiceId> |
| **dep:NSPCommunicationID**<br>*<minOccurs="0">* | Identification of the message assigned by the NSP. It must have the following format: [NSP name] + [NSP Gateway ID] + [date time] + [message sequence number].<br>**Restriction** is set on base type "**string [100]**". | <dep:NSPCommunicationId><br>nsp-name1.gtw134567.<br>20100908185555.123456<br></dep:NSPCommunicationId> |
| **dep:T2SMessageId**<br>*<minOccurs="0">* | Identification of the message sent by T2S Platform.<br>**Restriction** is set on base type "**string [100]**". | <dep:T2SMessageId><br>T2S.MSGRT.NSPname1.2011<br>0101000000.000001<br></dep:T2SMessageId> |
| **dep:T2SActorMessageId**<br>*<minOccurs="0">* | Unique message identifier generated at Directly Connected T2S Actor site.<br>**Restriction** is set on base type "**string [100]**". | <dep:T2SActorMessageId><br>T2SActorGateway1.20100908<br>175531.123456<br></dep:T2SActorMessageId> |
| **dep:EntryTimestamp**<br>*<minOccurs="0">* | Timestamp of the NSP's gateway reception.<br>**Restriction** is set on base type "**dateTime**". | <dep:EntryTimestamp><br>2011-01-01T00:00:00<br></dep:EntryTimestamp> |
| **dep:SendTimestamp**<br>*<mandatory tag>* | Timestamp of the sending of message.<br>**Restriction** is set on base type "**dateTime**". | <dep:SendTimestamp><br>2011-01-01T00:00:00<br></dep:SendTimestamp> |
| **dep:ReceiveTimestamp**<br>*<mandatory tag>* | Timestamp of the receiving of message.<br>**Restriction** is set on base type "**dateTime**". | |
| **dep:PDMHistory**<br>*<mandatory tag>* | History of the deliveries of the message/file in case that the message was already sent but not correctly acked.<br>**Restriction** is set on base type "**dateTime**". | <dep:PDMHistory><br>2011-01-01T00:00:00<br>2011-02-14 T00:10:01<br></dep:PDMHistory> |

| TAG NAME | TAG DESCRIPTION / ALLOWED VALUES | EXAMPLE |
|---|---|---|
| **dep:DeliveryMode**<br>*<mandatory tag>* | Identification of real time or store-and-forward exchange.<br>**Enumeration** with possible values:<br>• "**RT**" for Real-Time;<br>• "**SF**" for Store-and-forward. | <dep:DeliveryMode><br>RT<br></dep:DeliveryMode> |
| **dep:DeliveryNotification**<br>*<minOccurs="0">* | Delivery notification management; this field has to be set only in the case of **store-and-forward mode**.<br>The following values are foreseen:<br>• "**YES**": the delivery notification is requested always<br>• "**FAIL**": the delivery notification is requested only in case of failure<br>• "**NO**": the delivery notification is not requested<br>**Restriction** is set on base type "**string**". | <dep:DeliveryNotification><br>FAIL<br></dep:DeliveryNotification> |
| **dep:**<br>**NonRepudiationExchange**<br>*<minOccurs="0">* | Flag that indicates that a non-repudiation on exchange is requested (or not) for this message/file exchange.<br>**Enumeration** with possible values: **YES** or **NO**. | |
| **dep:Encryption**<br>*<minOccurs="0">* | Flag that indicates that the message/file is encrypted.<br>**Enumeration** with possible values: **YES** or **NO**. | |
| **dep:EncryptionAlgorithm** | Algorithm used for encryption. (only for future use) | |
| **dep:Compression**<br>*<minOccurs="0">* | Flag that indicates the algorithm used to compress the payload or NONE (if compression is not used).<br>**Enumeration** with fixed value "**NONE**". | |
| **dep:FileFormat**<br>*<minOccurs="0">* | Format of file (e.g. **ISO 20022**). This field has to be set only in the case of **file transfer service**, with TechnicalServiceID tag set to FILERT (Real Time File) or FILESNF (Store & Forward File).<br>**Restriction** is set on base type "**string [100]**". | |
| **dep:ExchangeStatus**<br>*<mandatory tag>* | Status of the exchange.<br>**Enumeration** with possible values:<br>• "**OK**" in the case of successful exchange<br>• "**KO**" in case of failure | <dep:ExchangeStatus><br>OK<br></dep:ExchangeStatus> |
| **dep:ErrorDescription**<br>*<minOccurs="0">* | Description of the error occurred during the exchanging process (tag has to be set only if tag dep:ExchangeStatus has value "KO" )<br>This is a **complex type** tag based on two elements:<br>• **ErrorCode**, a mandatory tag with base type **string** and a validation pattern "**T2S[0-9]{3}E**".<br>• **AdditionalInfo**, an optional tag with **restriction** set on base type "**string [200]**". | <dep:ErrorDescription><br><dep:ErrorCode><br>T2S040E<br></dep:ErrorCode><br><dep:AdditionalInfo><br>Message expired.<br>Receiver has not been<br>connected for 14 days.<br></dep:AdditionalInfo><br></dep:ErrorDescription> |
| **dep:MessageDigest**<br>*<minOccurs="0">* | Digest of the Message/File exchanged; the digest has to be applied to the full "Message Text" part of the WebSphere MQ message.<br>**Restriction** is set on base type "**string [1024]**". | |

It is worth mentioning specifically the dep:TechnicalServiceId tag. In the DEP, business data can be exchanged as a message or as a file. A message is a data structure containing a financial instruction or a piece of information based on the XML format, whereas a file is a data structure containing one or more messages in XML format.

Signature management is based on the XML Advanced Electronic Signature (XAdES) standard. In particular, the DEP adopts the XAdES-T format, including the timestamp. In case of dep:NonRepudiationExchange enabled:

- The sending counterpart (i.e. T2S) adds a signature by the end of the message based on the "Exchange Header " and the "Business Message" content.
- The receiving counterpart (i.e. the DCP) checks the validity of the signature and sends back an error messages (NAN Technical ack) in case of failure (Code T2S999E).

The channel through which data are exchanged, both for messages and files, defines the maximum size of the data structure of the DEP message. Size limits for each channel are summarized in TABLE 3.

**TABLE 3: SIZE LIMITS FOR THE VARIOUS CHANNELS.**

| | MINIMUM LENGTH | MAXIMUM LENGTH |
|---|---|---|
| Message channel | 0 | 32 KB |
| File channel | 0 | 32 MB |

## 4.5. Real-time files and messages exchange

The following sections describe the real-time protocol for files and messages exchange, both for inbound and outbound flows.

### 4.5.1. DEP for real-time inbound messages

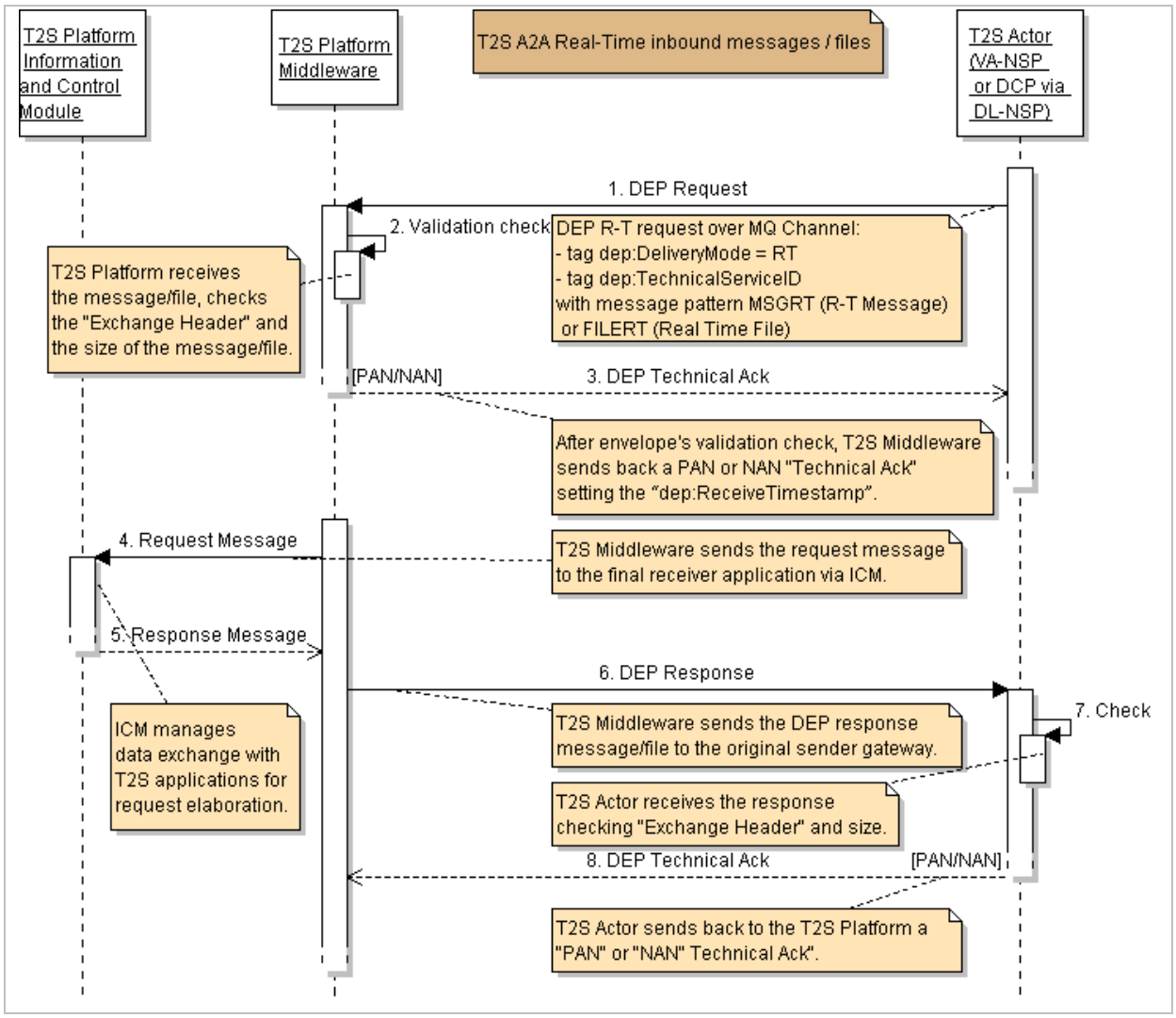
The sequence Diagram 7 shows a typical scenario for an A2A inbound real-time file/message exchange.

Following the message sequence shown in the diagram timeline, the table 4 below describes the main interactions between the T2S actor and the T2S platform. The assumption underlying this description is that the T2S actor is a DCP connected to T2S via a direct link.

**TABLE 4 - REAL-TIME T2S INBOUND MESSAGE – DESCRIPTION OF THE MAIN USE CASE**

| STEP NUMBER | STEP DESCRIPTION |
|---|---|
| 1) | The counterpart sends a real-time message/file to the T2S platform by a "Request" primitive.<br>The "Delivery Mode" field is set to "RT". The field T2SActorMessageId has to been set by DCP to the unique message identification generated at Directly Connected T2S Actor gateway site.<br>If the T2S actor is the VA-NSP, "dep:NSPCommunicationID" envelope field has to be generated (this identifier shall be unique at NSP level). |
| 2) | The T2S Platform receives the message/file and performs the validation check of the "Exchange Header" and checks the size of the message/file. After the validation of the envelope, the T2S Platform sends back to the DCP (or to the NSP's gateway) a PAN or NAN "Technical Ack" setting the "dep:ReceiveTimestamp" with the receiving time (MQMD.putime field of the WMQ message).<br>If a NAN is returned the flow is completed and the reason of the failure is set in the field dep:ErrorCode and dep:ErrorDescription of the response message's header (ref. alternative scenario modelled in Diagram 7).<br>As an alternative scenario, where T2S actor is a VA-NSP, if a NAN is returned the flow is completed and the NSP has to inform the counterpart about the failure. If the T2S Platform doesn't answer with a response in the timeout timeframe, the NSP shall send a "timeout" information to the sender. |
| 3) | The message/file is passed to the T2S application, interacting with T2s-ICM (Interface Control Module). |
| 4) | The T2S application passes the response to the T2S network interface component. |
| 5) | The T2S Platform sends the "response" message to the DCP (or NSP's gateway), setting in the "Exchange Header" the "dep:T2SMessageID" to a unique identifier and keeping all other fields as received in the "request" message. |
| 6) | The DCP (or the NSP's gateway) receives the "response" and performs the validation check of the "Exchange Header" part and of the size.<br>If the validation process fails, or the size of the response is not in the allowed range, then the T2S Actor sends back to the T2S Platform a "NAN Technical Ack" setting in appropriate way the "dep:ExchangeStatus" and "dep:ErrorDescription" fields. If T2S Actor is the NSP, it informs the sender about the failure with a response error message.<br>The flow is now completed. |

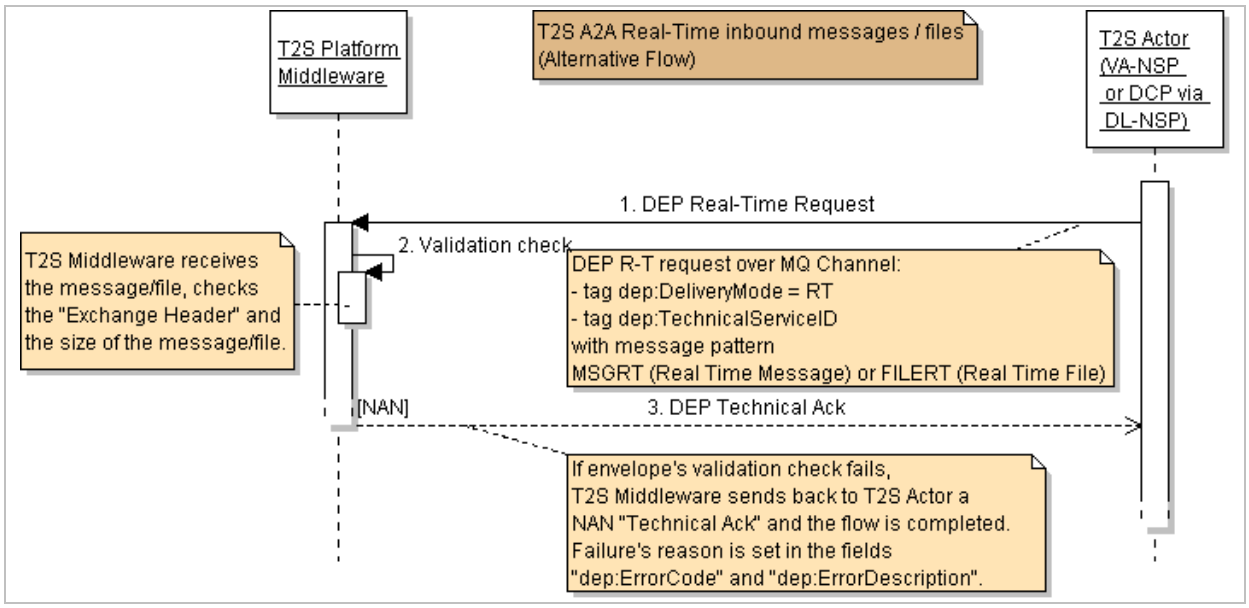
**DIAGRAM 7: REAL-TIME T2S INBOUND MESSAGE – SD OF THE MAIN USE CASE**




In a scenario where the DCP is connected to T2S via a VA-NSP, the role of the T2S actor in the sequence diagram is played by the gateway of the VA-NSP and the first step in diagram is a real-time message/file sent from the counterpart to the gateway of the NSP. In this case, if the size of the message/file is out of the allowed range, then the gateway of the NSP rejects the exchange and sends an error message back to the DCP.

As a final step, the gateway of the NSP sends the response to the counterpart, including the information of the T2SMessageID fields generated by T2S.

**DIAGRAM 8: REAL-TIME T2S INBOUND MESSAGE – ALTERNATIVE SD – NAN TECHNICAL ACK**

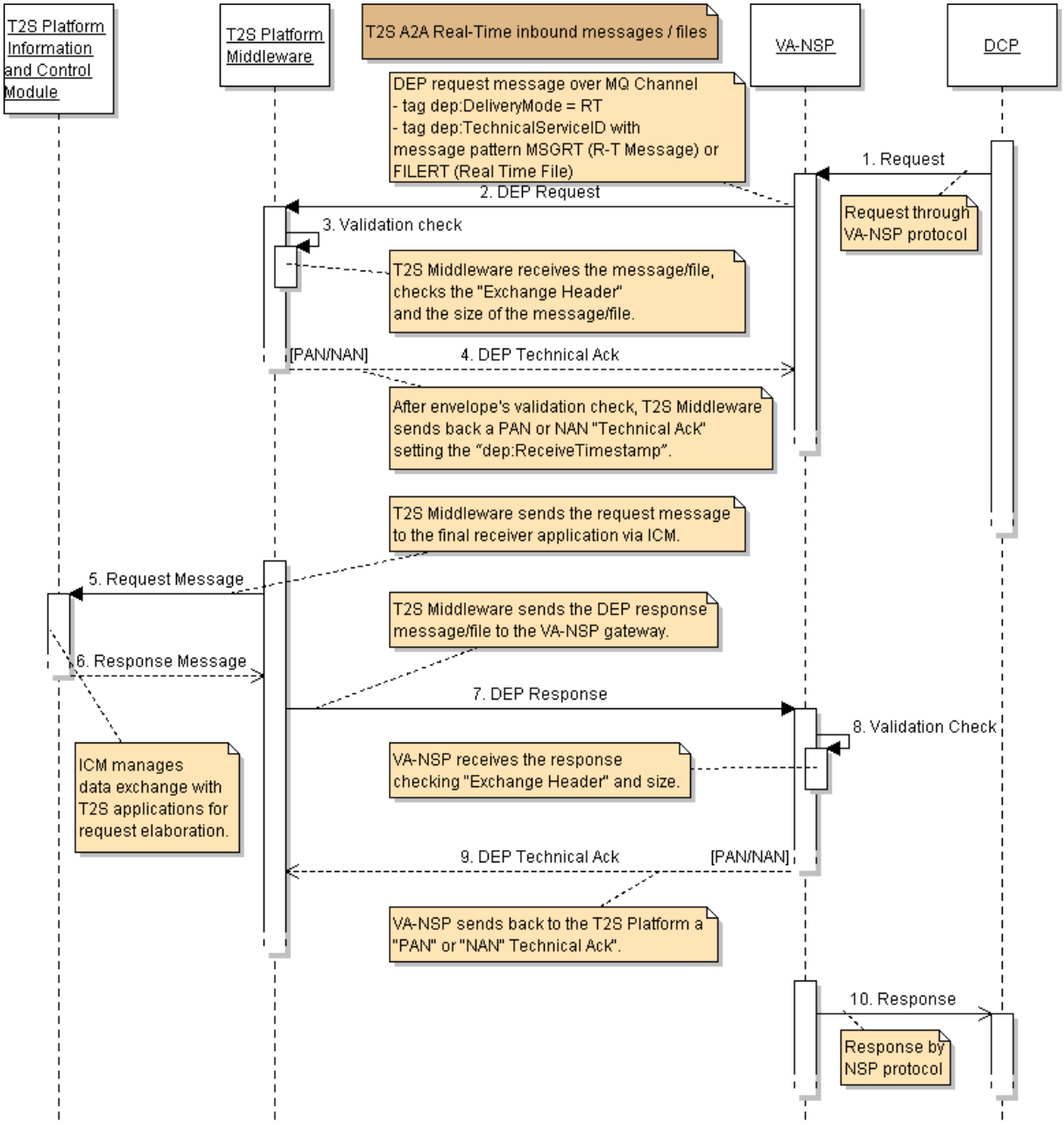



If T2S actor were the gateway of the VA-NSP, first step in diagram would be a real-time message/file sent from counterpart to NSP's gateway (reference to Diagram 9).

In this case, if the size of the message/file is outside of the allowed range, the NSP's gateway must reject the exchange with an error message sent to Directly Connected T2S Actor.

As the final step, the NSP's gateway sends the "response" to the counterpart including the information of the T2SMessageID fields generated at the T2S Site in the response message.

**Diagram 9 : Real-Time T2S inbound – SD of the main Use Case (VA-NSP scenario)**



### 4.5.2. DEP for real-time outbound messages

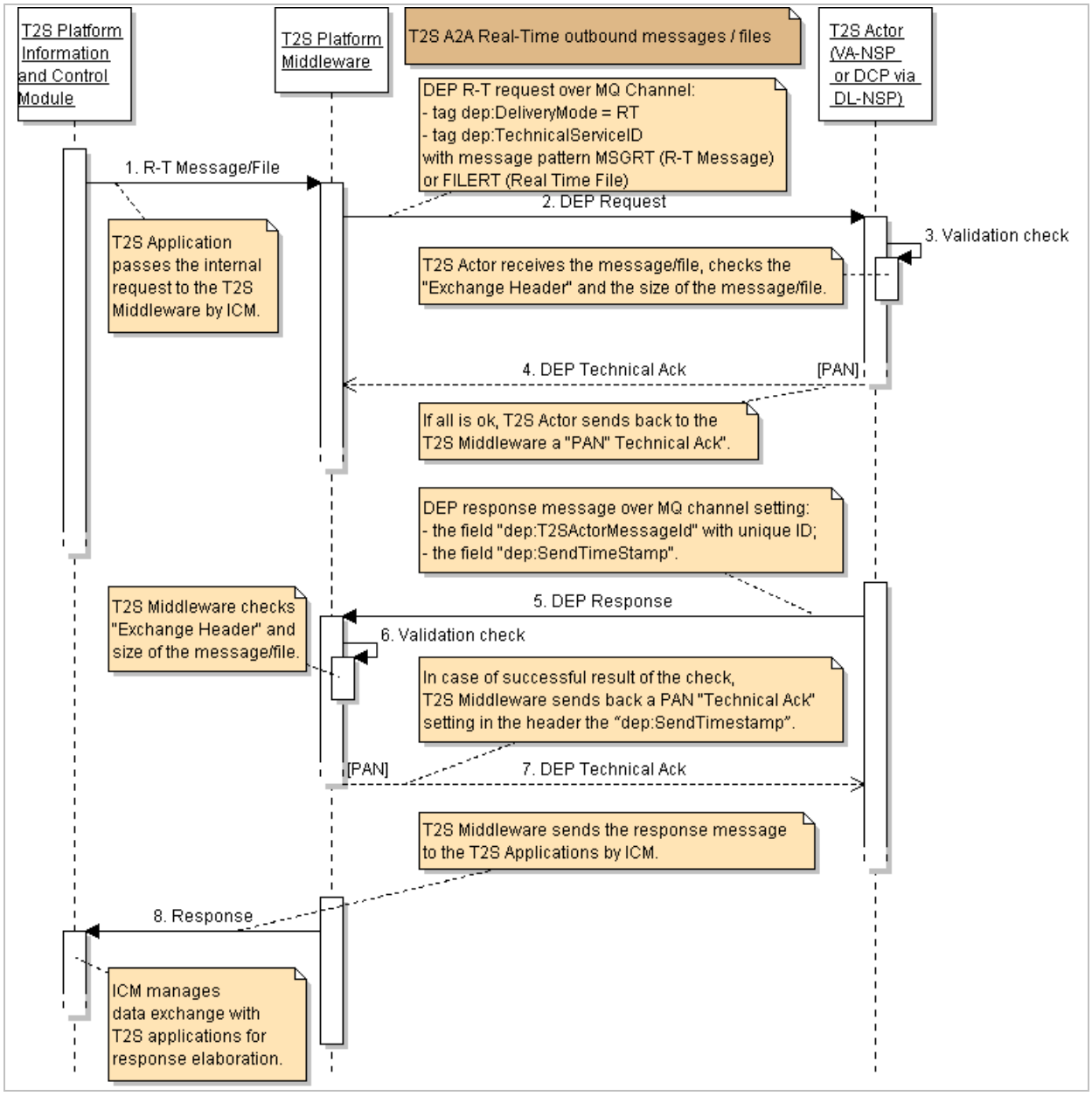
The sequence Diagram 10 shows a typical scenario for an A2A outbound real-time file/message exchange.

Following the message sequence shown in the diagram timeline, the following table describes the main interactions between the T2S platform and the T2S Actor. The assumption underlying this description is that the T2S actor is a DCP connected to T2S via a direct link.

**TABLE 5 - REAL-TIME T2S OUTBOUND MESSAGE – DESCRIPTION OF THE MAIN USE CASE**

| STEP NUMBER | STEP DESCRIPTION |
|---|---|
| 1) | The T2S application passes a real-time message/file to T2S network interface component, to be sent towards T2S Actor. |
| 2) | The T2S Platform sends a "Request" primitive to the T2S Actor. The "T2SMessageID" field has to be generated by the T2S Platform (this identifier shall be unique at T2S level). The "Delivery Mode" field is set to "RT". |
| 3) | The T2S Actor receives the message/file, performs the validation check of the "Exchange Header" part and checks the size of the message/file. |
| 4) | The DCP saves the message, assigns to it a unique identification, stores this value in the "dep:T2SActorMessageId" field of the "Exchange Header", saves the current timestamp in the "dep:EntryTimestamp" field and sends back to T2S Platform the "PAN Technical Ack".<br>If the actor is a VA-NSP, unique identification is stored in the "dep:NSPcommunicationID" field of the "Exchange Header".<br>As an **alternative scenario**, if the validation process fails, then the T2S Actor sends back to T2S Platform a "NAN Technical Ack" setting the error description field with the reason of the failure and the flow is completed. |
| 5) | In the case of a successful result of the check the receiver sends the "response" to the T2S Platform setting in the "Exchange Header" a unique identification of the response generated at receiver site in the dep:T2SActorMessageId. |
| 6) | The T2S Platform performs the "Exchange Header" validation and checks the size of the message coming from the receiver. |
| 7) | The T2S Platform sends back to the T2S Actor a PAN or NAN "Technical Ack".<br>If the size is outside of the allowed range the message is rejected and an error message is returned to the receiver: in this case a response message is sent to the T2S Actor with the "dep:ErrorDescription" field set to "ERROR occurred – Message/File exchange aborted" value.<br>In case of a successful result of the check the T2S Platform sends back the PAN setting in the "Exchange Header":<br>• the field "dep:T2SActorMessageId" with the unique identification generated at client site;<br>• the field "dep:SendTimestamp" with the time of the sending time (in reference to point 4) |
| 8) | T2S Network Interface Component sends the response to T2S Application. |

**DIAGRAM 10: REAL-TIME T2S OUTBOUND MESSAGE – SD OF THE MAIN USE CASE**

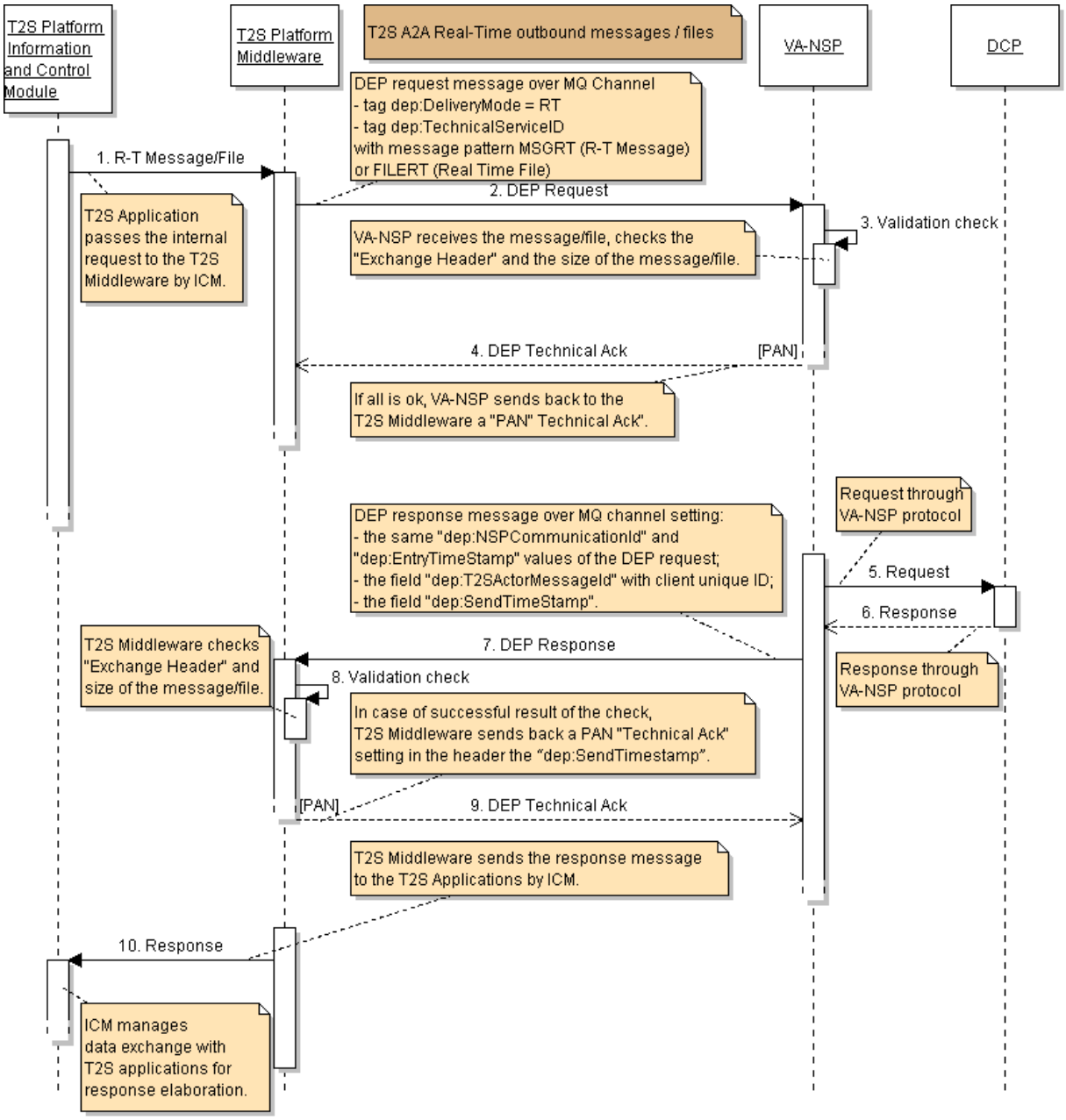



If the T2S Actor is a VA-NSP, after having received a PAN technical Ack (step 3), the VA-NSP sends the message to the final receiver (i.e. the DCP). If there is an error in the transmission to the final receiver (for instance the receiver is not connected), or if the transmission is not completed before the timeout, then the gateway of the NSP sends back to the T2S platform a response message with the "dep:ExchangeStatus" field set to "**KO**" and the "dep:ErrorDescription" field set with the reason code of the error occurred and the flow is completed. The business part of the response message in case of an error is not included in the message.

In case of successful result, the receiver sends back the response to the gateway of the NSP, setting in the message header a unique identification of the response generated by the receiver site. Then, the gateway of the NSP checks the size of the message sent by the receiver. If the size is out of the allowed range, then the message is rejected and an error message is returned to the receiver. In this case, the T2S platform receives a response message with the "dep:ErrorDescription" field set to "ERROR occurred – Message/File exchange aborted".

In case of successful check, the NSP sends the response to the T2S platform as described in table 5, from step 5.

**Diagram 11: Real-Time T2S outbound – SD of the main Use Case (VA-NSP scenario)**



## 4.6. Store-and-forward files and messages exchange

The following sections describe the store-and-forward protocol for files and messages exchange, both for inbound and outbound flows.

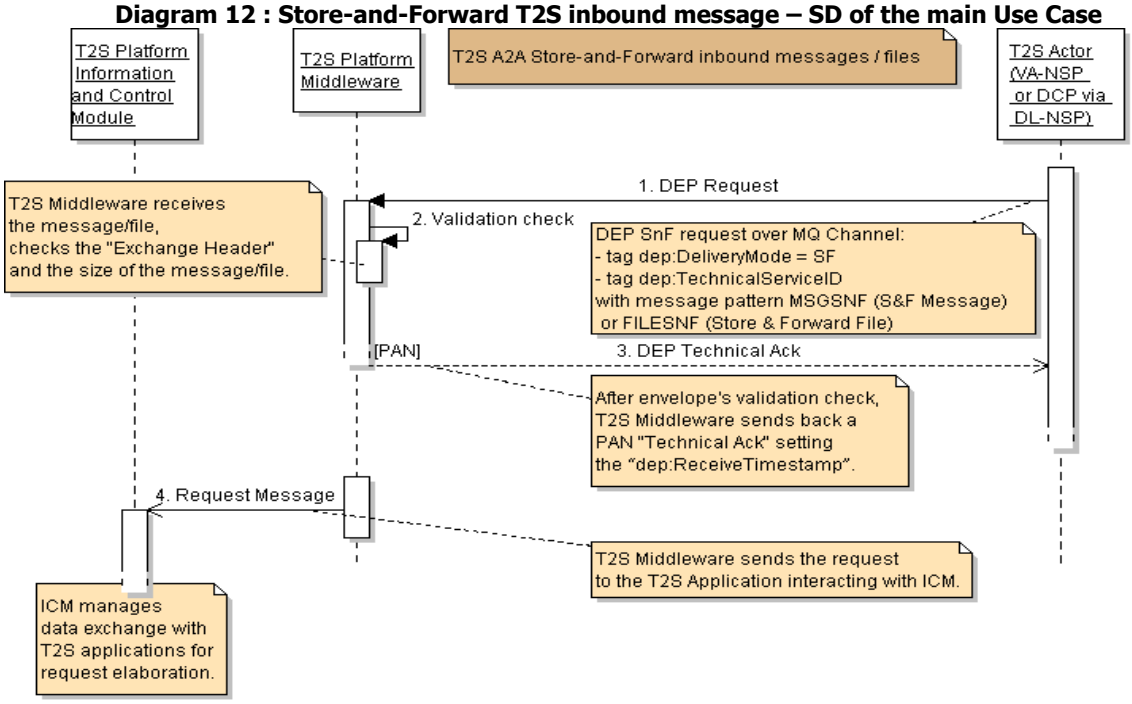
### 4.6.1. DEP for store-and-forward inbound messages

Following the message sequence shown in the diagram timeline, TABLE 6 below describes the main interactions between the T2S actor and the T2S platform. The assumption underlying this description is that the T2S actor is a DCP connected to T2S via a direct link.

**TABLE 6: STORE & FORWARD T2S INBOUND MESSAGE – DESCRIPTION OF THE MAIN USE CASE**

| STEP NUMBER | STEP DESCRIPTION |
|---|---|
| 1) | If the T2S Platform has enabled the store-and-forward traffic, the T2S Actor sends the message/file to the T2S Platform gateway. |
| 2) | The T2S Platform receives the message/file and performs the validation check of the "envelope" part. In the case that the T2S Platform doesn't send the Technical Ack within 10 minutes the NSP shall manage the condition as described in the requirement T2S.UC.TC.30195. |
| 3) | After the validation check, the T2S Platform sends back to the T2S Actor a PAN or NAN "Technical Ack" setting the "**dep:ReceiveTimestamp**" and the "**dep:T2SMessageId**" generated for the incoming message with the receiving time (MQMD.putime field of the WMQ message).<br>If a NAN is returned in VA-NSP scenario, the NSP shall retry for up to 10 times the delivery after which a delivery failure notification is send back to the Directly Connected T2S Actor and the flow is completed.<br>In case of non repudiation scenario, after the validation check the T2S Platform sends back to the T2S Actor a signed "Technical Ack"; the signature will be appended at the end of the ExchangeHeader. |
| 4) | The message is passed to the T2S application. |

**Diagram 12 : Store-and-Forward T2S inbound message – SD of the main Use Case**



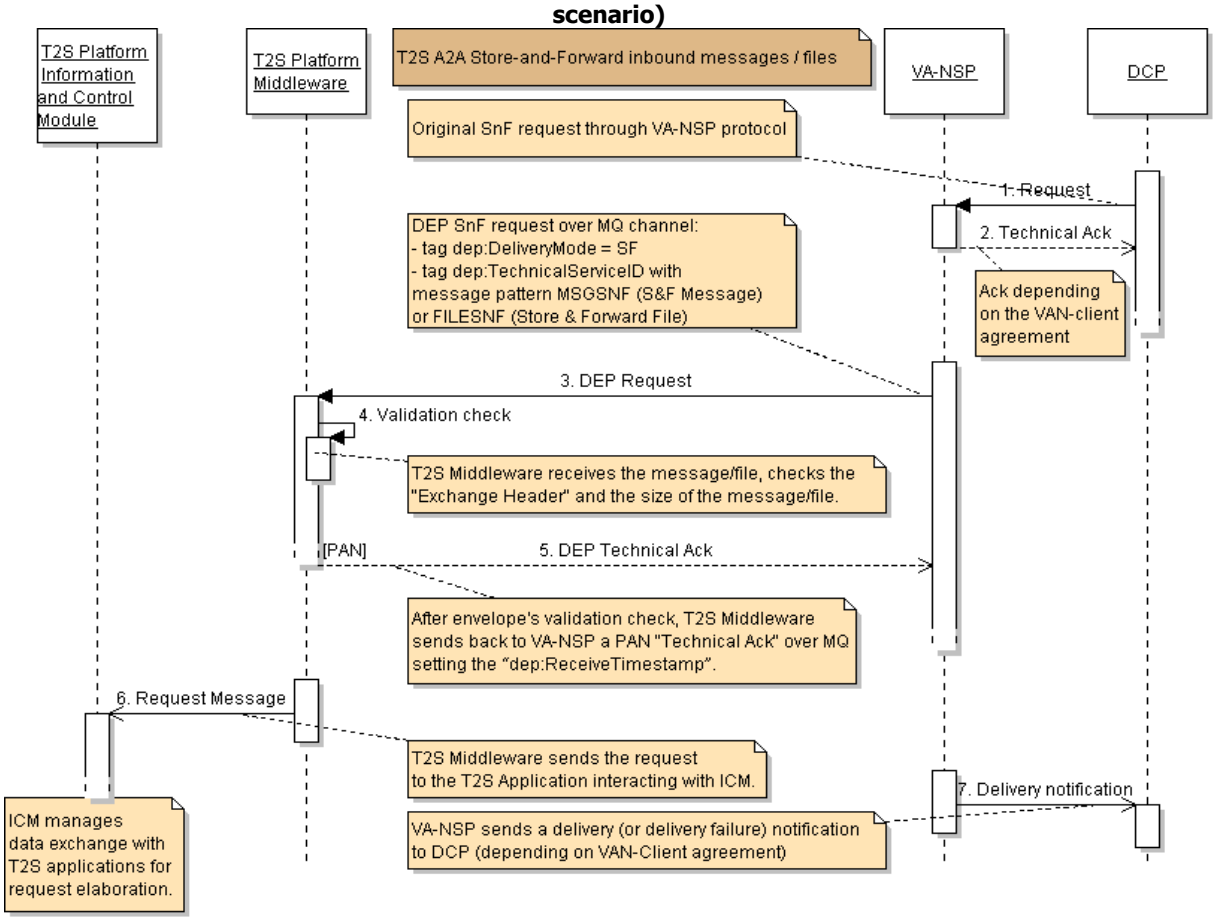

In a scenario where the DCP is connected to T2S via a VA-NSP (i.e. the original sender is a DCP connected to T2S via a VA-NSP) the process requires the following additional steps:

1. The DCP sends the original request message/file to the gateway of the VA-NSP;
2. The gateway of the VA-NSP sends a Technical Ack back to the DCP, after having performed a check on the size of the message/file (and rejecting it in case of failure)[2];
3. The processing takes places following the steps from 1 to 4 described for the typical scenario above;
4. Depending on the T2S connectivity service agreement between the NSP and the DCP, the NSP sends to the DCP a delivery (or a delivery failure) notification including the timestamp of the reception set by the T2S platform in the "dep:ReceiveTimestamp" field mentioned above (step 3 of the typical scenario). N.B.: This delivery notification (or delivery notification failure) from VA-NSP to DiCoA is optional.

Following sequence diagram (diagram 13) shows the overall data exchange flow where T2S DiCoA communicates with T2S platform by VA-NSP.

[2] The sending of this Technical Ack from the VA-NSP to the DCP depends on the agreement between the VA-NSP and the DCP.

**Diagram 13: Store-and-Forward T2S inbound – SD of the main Use Case (VA-NSP scenario)**




## 4.6.2. DEP for store-and-forward outbound messages

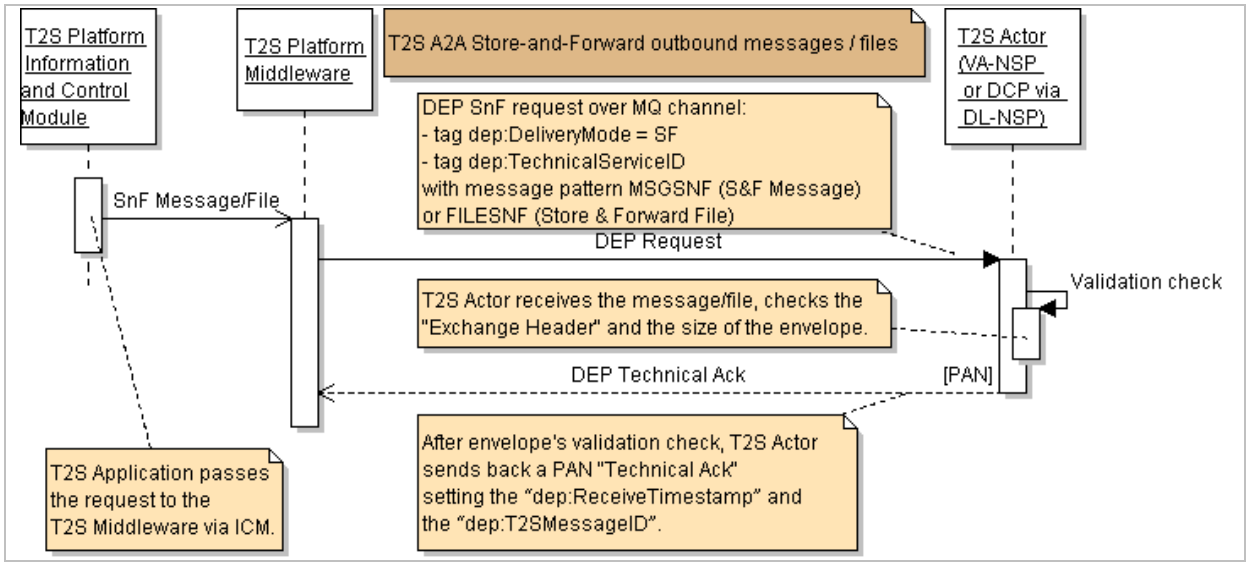
When the T2S Platform needs to send a message/file in store-and-forward mode to its clients, it will go through an **Store-and-Forward (SnF) outbound scenario**. The **main scenario of a A2A store-and-forward file/message exchange**, for a **T2S outbound**, is modeled in Diagram 14.

Following the message order showed by diagram timeline, Table 7 describes the main interactions between T2S Platform and T2S Actor. There, we assume T2S Actor to be a DiCoA which communicates by direct link, without VA-NSP intermediation.

**Table 7: Store & Forward T2S outbound message – Description of the main Use Case**

| STEP NUMBER | STEP DESCRIPTION |
|---|---|
| 1) | The T2S Application passes the request to the T2S Network Interface Component. |
| 2) | The T2S Platform sends a "Request" message/file primitive to the T2S Actors. The "T2S Message Id" envelope field is generated by the T2S Platform (this identifier shall be unique at T2S level). The "Delivery Mode" field is set to "SF". |
| 3) | The T2S Actor receives the message/file and performs the validation check of the "Envelope" part and the validation of the size of the message/file. |
| 4) | If the validation check is passed, the T2S Actor sends back to T2S Platform a "PAN Technical Ack" setting "**dep:T2SActorMessageId**" , "dep:ReceiveTimestamp" and "dep:T2SMessageId" with values generated for the incoming message with the receiving time (MQMD.putime field of the WMQ message).<br>In case of signed technical ack, for non repudiation, the signature will be appended at the end of the ExchangeHeader.<br>If the validation process fails, the T2SActor sends back to the T2S Platform a "NAN Technical Ack" setting in the "dep:ExchangeStatus" and "dep:ErrorDescription" fields appropriately; the flow is so completed. |

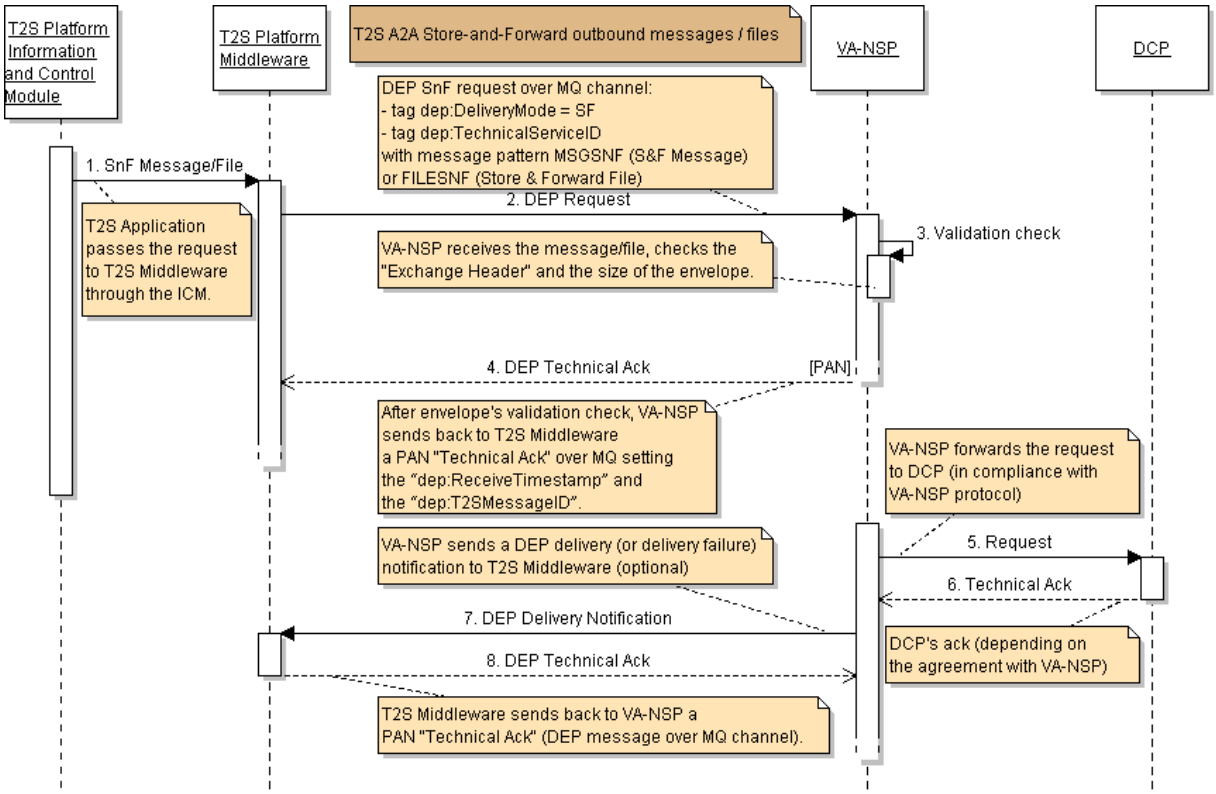
**Diagram 14: Store-and-Forward T2S outbound message – SD of the main Use Case**




In a scenario where the DCP is connected to T2S via a VA-NSP (i.e. the final receiver is a DCP connected to T2S via a VA-NSP) the process requires some additional steps. After step 4 of the typical scenario (PAN ack from the VA-NSP to the T2S platform), if the receiving DCP is available for store-and-forward traffic, the gateway of the NSP sends the message/file to it. Then, the receiving DCP sends back to the gateway of the NSP a Technical Ack (if and according to the format agreed between the VA-NSP and the DCP). Finally, the gateway of the NSP sends a DeliveryNotify message (including the same NSP communication id of the original request) back to the T2S Platform and it receives a Technical Ack from the T2S network interface component.

Following sequence diagram (Diagram 15) shows the overall data exchange flow where DCP communicates with T2S platform by VA-NSP.

**Diagram 15: Store-and-Forward T2S outbound – SD of the main Use Case (VA-NSP scenario)**




As an alternative scenario, if the delivery of message/file has failed for 10 times when the receiver is available, or the Directly Connected T2S Actor is unavailable for store-and-forward traffic for 14 calendar days, the NSP's gateway sends back to the T2S Platform a "DeliveryNotify" message with the same "NSP communication id" of the original request and with the information of the error occurred on the delivery.

If in the original request the "dep:DeliveryNotification" field was set to "YES", the NSP shall send a "DeliveryNotify" message also in successful condition.

When the "DeliveryNotify" is received by the T2S Platform, it sends a Technical Ack back to the NSP and the flow is completed.

## 4.7. Controlling the store-n-forward traffic

The T2S Operator can enable/disable data exchange in store-n-forward mode during the maintenance window or for contingency reason.

When the T2S platform is ready to manage the store-and-forward traffic, it sends an EnableSnfTraffic message to the gateway of the NSP. This is a control message and it does not foresee the standard envelope used for business messages exchange. Section "Echange Header XSD and message examples" provides an example of this message.

## 4.8. Processing functionalities for incoming A2A messages

This section describes the functions provided by the middleware component for incoming A2A messages.

### 4.8.1. Exchange Header validation

This function checks the Exchange Header of DEP messages against format its schema definition. Section "Echange Header XSD and message examples" provides the XML Schema Definition (XSD) of the Exchange Header.

In addition, the T2S platform performs the following validations on the Exchange Header:

- It checks that the connected user complies with the user specified in the Sender: field of the ExchangeHeader;
- It checks that the TechnicalServiceId specified in the Exchange Header corresponds to the incoming queue used by the DCP (this includes the "logical environment" selected control).

After the validation of the Exchange Header, the T2S platform sends a PAN or a NAN Technical Ack back to the DCP, setting the dep:ReceiveTimestamp to the actual receiving time (MQMD.putime field of the WMQ message).

### 4.8.2. Technical ack/nak management

The T2S platform provides a Technical Ack for each data exchange with a NSP or a DCP, for confirmation of the data exchange completion.

The NSP manages the technical acknowledgement process as follows. The Technical Ack is a WebSphere MQ report message with type PAN (Positive Application Notification) or NAN (Negative Application Notification). The receiving WebSphere MQ application (i.e. the T2S middleware component or the gateway of the NSP) send this report back when undertaking a message (i.e. when the message is stored or processed). Section 4.2 - MQMD message format describes the structure of the Technical Ack.

In case of store-and-forward communication, the NSP forwards the full content of the "Message Text" part of the MQ message to the original sender in the delivery notification.

The Technical Ack is returned to the sender (i.e. to the T2S platform or to the NSP) within a time-frame of 10 minutes[3].

For real-time communication, no specific actions are required for timing issues, owing to the time-out management mechanism. For store-and-forward incoming messages, in case that the time-out for the Technical Ack is exceeded, the NSP re-sends the message including in the ExchangeHeader section the "dep:PDMHistory" element specifying the delivery time of the previous attempt(s) in the following format:

[3] This value can be configured.

```
<dep:PDMHistory>
       2011-01-01T00:00:00
       2011-02-14 T00:10:01
</dep:PDMHistory>
```

After 10 unsuccessful attempts, the NSP sends a Delivery Notification Failure back to the original sender and it suspends the sending of the store-and-forward messages/files to the T2S platform. An alarm is triggered in order to allow to the NSP staff to inform the T2S Operator that a problem occurred in the store-and-forward channel.

The NSP and the T2S platform manage the negative message acknowledgement in all cases of error. In this case, the originator of the message receives a NAN. The "dep:ExchangeStatus" field is set to the value "KO" and the "dep:ErrorDescription" field is set according to the following table.

**TABLE 8: T2S ERROR CODING**

| CODE | ERROR OCCURRED | ERROR DESCRIPTION FIELD VALUE |
|---|---|---|
| **T2S010E** | The message or file size is not in the allowed range | "Message or file size is not in the allowed size range" |
| **T2S020E** | One (or more) fields are not well formed | "Field xxxx not well formed" |
| **T2S030E** | One (or more) mandatory fields of Exchange Header are not present | "Field xxxx missing" |
| **T2S040E** | Timeout condition | "Timeout occurred" |
| **T2S999E** | All other errors | "ERROR occurred – Message/File exchange aborted" |

## 4.9. Echange Header XSD and message examples

Exchange header XSD

```
<?xml version="1.0" encoding="UTF-8"?>
<schema       targetNamespace="http://www.ecb.eu/t2s-0.2"       elementFormDefault="qualified"
xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:t2s="http://www.ecb.eu/t2s-0.2">

  <simpleType name="DistinguishedNameType">
    <restriction base="string">
      <maxLength value="100"/>
    </restriction>
  </simpleType>

  <simpleType name="TechnicalServiceIdType">
    <restriction base="string">
      <pattern value=".+(MSGRT|MSGSNF|FILERT|FILESNF).(EAC|UTEST|PROD)"/>
      <maxLength value="60"/>
    </restriction>
  </simpleType>

  <simpleType name="NSPCommunicationIDType">
    <restriction base="string">
      <maxLength value="100"/>
    </restriction>
  </simpleType>

  <simpleType name="T2SMessageIdType">
    <restriction base="string">
      <maxLength value="100"/>
    </restriction>
  </simpleType>

  <simpleType name="DeliveryModeType">
    <restriction base="string">
      <enumeration value="RT"/>
```

```
    <enumeration value="SF"/>
  </restriction>
</simpleType>

<simpleType name="CompressionIndicatorType">
  <restriction base="string">
    <enumeration value="NONE"/>
  </restriction>
</simpleType>

<simpleType name="FileFormatType">
  <restriction base="string">
    <maxLength value="100"/>
  </restriction>
</simpleType>

<simpleType name="RetentionPeriodType">
  <restriction base="int">
    <maxInclusive value="14"/>
    <minInclusive value="1"/>
    <whiteSpace value="collapse"/>
  </restriction>
</simpleType>

<simpleType name="TimestampType">
  <restriction base="dateTime"/>
</simpleType>

<simpleType name="ServiceNameType">
  <restriction base="string">
    <maxLength value="50"/>
  </restriction>
</simpleType>

<simpleType name="SnFQueueManagerNameType">
  <restriction base="string">
    <maxLength value="48"/>
  </restriction>
</simpleType>

<simpleType name="SnFQueueNameType">
  <restriction base="string">
    <maxLength value="48"/>
  </restriction>
</simpleType>

<simpleType name="SnFStatusType">
  <restriction base="string">
    <enumeration value="Failed"/>
    <enumeration value="Activated"/>
    <enumeration value="Deactivated"/>
  </restriction>
</simpleType>

<simpleType name="NonRepudiationType">
  <restriction base="string">
    <enumeration value="Yes"/>
    <enumeration value="No"/>
  </restriction>
</simpleType>

<simpleType name="EncryptionType">
  <restriction base="string">
    <enumeration value="Yes"/>
    <enumeration value="No"/>
  </restriction>
</simpleType>

<simpleType name="ReasonType">
  <restriction base="string">
    <maxLength value="100"/>
  </restriction>
</simpleType>
```

```
  <simpleType name="ErrorCodeType">
    <restriction base="string">
      <pattern value="T2S[0-9]{3}E"></pattern>
    </restriction>
  </simpleType>

  <simpleType name="AdditionalInfoType">
   <restriction base="string">
    <maxLength value="200"></maxLength>
   </restriction>
  </simpleType>

  <simpleType name="VersionType">
    <restriction base="string">
      <enumeration value="0.2"></enumeration>
    </restriction>
  </simpleType>

  <simpleType name="T2SActorMessageIdType">
    <restriction base="string">
      <maxLength value="100"></maxLength>
    </restriction>
  </simpleType>

   <simpleType name="MessageDigestType">
    <restriction base="string">
      <maxLength value="1024"></maxLength>
    </restriction>
  </simpleType>


  <simpleType name="ExchangeStatusType">
    <restriction base="string"></restriction>
  </simpleType>

  <complexType name="SnFServiceType">
    <sequence>
      <element name="Name" type="t2s:ServiceNameType"/>
      <element name="DestQmanagerName" type="t2s:SnFQueueManagerNameType"/>
      <element name="DestQueueName" type="t2s:SnFQueueNameType"/>
    </sequence>
  </complexType>

  <complexType name="SnFServiceAckType">
    <complexContent>
      <extension base="t2s:SnFServiceType">
        <sequence>
          <element name="Status" type="t2s:SnFStatusType"/>
          <element name="Reason" type="t2s:ReasonType" minOccurs="0" maxOccurs="1"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>

  <complexType name="SnFTrafficCommandType">
    <sequence>
      <element          name="Service"          type="t2s:SnFServiceType"          minOccurs="1"
maxOccurs="unbounded"></element>
    </sequence>
  </complexType>

  <complexType name="SnFTrafficCommandAckType">
    <sequence>
      <element         name="Service"         type="t2s:SnFServiceAckType"         minOccurs="1"
maxOccurs="unbounded"></element>
    </sequence>
  </complexType>

  <complexType name="BusinessApplicationHeaderType">
    <complexContent>
      <extension base="anyType"></extension>
    </complexContent>
```

```
  </complexType>

  <complexType name="BusinessMessageType">
    <complexContent>
      <extension base="anyType"></extension>
    </complexContent>
  </complexType>

  <complexType name="ErrorDescriptionType">
    <sequence>
      <element name="ErrorCode" type="t2s:ErrorCodeType"></element>
      <element name="AdditionalInfo" minOccurs="0" type="t2s:AdditionalInfoType">
      </element>
    </sequence>
  </complexType>

  <complexType name="ExchangeHeaderType">
    <annotation>
      <documentation>Unique message identifier generated at T2S actor site</documentation>
    </annotation>
    <sequence>
      <element name="Version" type="t2s:VersionType">
        <annotation>
          <documentation>Version of Data Exchange Protocol</documentation>
        </annotation>
      </element>
      <element name="Sender" type="t2s:DistinguishedNameType">
        <annotation>
          <documentation>NSP user identification for T2S System User that send the
message</documentation>
        </annotation>
      </element>
      <element name="OriginalSender" type="t2s:DistinguishedNameType" minOccurs="0"></element>
      <element name="Receiver" type="t2s:DistinguishedNameType">
        <annotation>
          <documentation>NSP user identification for T2S System User that receive the
message</documentation>
        </annotation>
      </element>
      <element name="FinalReceiver" type="t2s:DistinguishedNameType" minOccurs="0"></element>
      <element name="TechnicalServiceId" type="t2s:TechnicalServiceIdType">
        <annotation>
          <documentation>
            Name of the service used to send messages and files

            NSP Name+ "." + &lt;msg-pattern&gt; + "." + &lt;environment&gt;

            where &lt;msg-pattern&gt; is one of: MSGRT MSGSNF FILERT FILESNF

            and &lt;environment&gt; is one of: EAC UTEST PROD
          </documentation>
        </annotation>
      </element>
      <element name="NSPCommunicationId" type="t2s:NSPCommunicationIDType" minOccurs="0"
maxOccurs="1">
        <annotation>
          <documentation>
            Identification of the message assigned by the NSP. It must have the following
format:

            NSP name + NSPGatewayId + &lt;datetime&gt; + &lt;msg-sequence-number&gt;
          </documentation>
        </annotation>
      </element>
      <element name="T2SMessageId" type="t2s:T2SMessageIdType" minOccurs="0">
        <annotation>
          <documentation>Identification of the message set by T2S Platform</documentation>
        </annotation>
      </element>
      <element name="T2SActorMessageId" minOccurs="0" type="t2s:T2SActorMessageIdType">
        <annotation>
          <documentation>Unique message identifier generated at T2S actor site</documentation>
        </annotation>
```

```
      </element>
      <element name="EntryTimestamp" type="t2s:TimestampType" minOccurs="0">
        <annotation>
          <documentation>Timestamp of the NSP's gateway reception</documentation>
        </annotation>
      </element>
      <element name="SendTimestamp" type="t2s:TimestampType">
        <annotation>
          <documentation>Timestamp of the sending of message</documentation>
        </annotation>
      </element>
      <element name="ReceiveTimestamp" type="t2s:TimestampType">
        <annotation>
          <documentation>Timestamp of the receiving of message</documentation>
        </annotation>
      </element>

      <element name="PDMHistory" type="t2s:TimestampType">
        <annotation>
          <documentation>Timestamp of the attempting of the delivery of the message
</documentation>
        </annotation>
      </element>

      <element name="DeliveryMode" type="t2s:DeliveryModeType">
        <annotation>
          <documentation>Identification of real time or store-and-forward
exchange</documentation>
        </annotation>
      </element>
      <element name="DeliveryNotification" type="string" minOccurs="0">
        <annotation>
          <documentation>
            Delivery notification management

            This field has to be set only in the case of store-and-forward mode. The following
values are
            foreseen: YES: the delivery notification is requested always FAIL: the delivery
notification is
            requested only in case of failure NO: the delivery notification is not requested
          </documentation>
        </annotation>
      </element>
      <element name="NonRepudiationExchange" type="t2s:NonRepudiationType" minOccurs="0"
maxOccurs="1">
        <annotation>
          <documentation>
            Flag that indicates that the non-repudiation is requested or not
          </documentation>
        </annotation>
      </element>

      <element name="Encryption" type="t2s:EncryptionType" minOccurs="0" maxOccurs="1">
        <annotation>
          <documentation>
            Flag that indicates that the message is encrypted or not
          </documentation>
        </annotation>
      </element>
      <element name="Compression" type="t2s:CompressionIndicatorType" minOccurs="0"
maxOccurs="1">
        <annotation>
          <documentation>
            Flag that indicates the algorithm used to compress the payload or NONE (if
compression is not
            used)
          </documentation>
        </annotation>
      </element>

      <element name="FileFormat" minOccurs="0">
        <annotation>
          <documentation>Format of file (e.g. ISO20022 or ISO…)</documentation></annotation>
```

```
        <simpleType>
          <restriction base="string">
            <maxLength value="100"></maxLength>
          </restriction>
        </simpleType>
      </element>
      <element name="ExchangeStatus">
        <annotation>
          <documentation>
            Status of the exchange
            "OK" in the case of a successful exchange "KO" in case of failure
          </documentation>
        </annotation>
        <simpleType>
          <restriction base="t2s:ExchangeStatusType">
            <enumeration value="OK"></enumeration>
            <enumeration value="KO"></enumeration>
          </restriction>
        </simpleType>
      </element>
      <element     name="ErrorDescription"     type="t2s:ErrorDescriptionType"     minOccurs="0"
maxOccurs="1">
        <annotation>
          <documentation>Description     of     the     error     occurred     during     the
exchanging</documentation>
        </annotation>
      </element>
      <element name="MessageDigest" type="t2s:MessageDigestType" minOccurs="0">
        <annotation>
          <documentation>Digest of the Message/File exchanged (The digest has to be applied to
the full Message Text part of the WebSphere MQ messages)</documentation>
        </annotation>
      </element>
    </sequence>
  </complexType>

  <complexType name="BusinessEnvelopeType">
    <sequence>
      <element                                                    name="BusinessApplicationHeader"
type="t2s:BusinessApplicationHeaderType"></element>
      <element name="BusinessMessage" type="t2s:BusinessMessageType"></element>
    </sequence>
  </complexType>

  <complexType name="ExchangeEnvelopeType">
    <sequence>
      <element name="ExchangeHeader" type="t2s:ExchangeHeaderType"></element>
      <element name="BusinessEnvelope" type="t2s:BusinessEnvelopeType"></element>
    </sequence>
  </complexType>

  <complexType name="TechnicalAckType">
    <sequence>
      <element
        name="ExchangeHeader"
        type="t2s:ExchangeHeaderType">
      </element>
    </sequence>
  </complexType>

  <complexType name="DeliveryNotificationType">
   <sequence>
    <element
     name="ExchangeHeader"
     type="t2s:ExchangeHeaderType">
    </element>
   </sequence>
  </complexType>

  <element name="Request" type="t2s:ExchangeEnvelopeType"/>

  <element name="Response" type="t2s:ExchangeEnvelopeType"/>
```

```
  <element name="EnableSnfTraffic" type="t2s:SnFTrafficCommandType"/>

  <element name="EnableSnfTrafficAck" type="t2s:SnFTrafficCommandAckType"/>

  <element name="DisableSnfTraffic" type="t2s:SnFTrafficCommandType"/>

  <element name="DisableSnfTrafficAck" type="t2s:SnFTrafficCommandAckType"/>

  <element name="TechnicalAck" type="t2s:TechnicalAckType"></element>

  <element name="DeliveryNotification" type="t2s:DeliveryNotificationType"></element>
</schema>
```

Message sent by T2S Platform to the NSP's gateway or DCP at the step no. 2:

```
<?xml version="1.0" encoding="UTF-8" ?>
<dep:Request
 xmlns:dep="http://www.ecb.eu/t2s-0.2"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.ecb.eu/t2s-0.2 dep-02.xsd ">
 <dep:ExchangeHeader>
  <dep:Version>0.2</dep:Version>
  <dep:Sender>cn=t2sappl1,o=t2sprod</dep:Sender>
  <dep:Receiver>cn=t2s-cust1,o=nsp-name1</dep:Receiver>
  <dep:TechnicalServiceId>nsp-name1.MSGRT.PROD</dep:TechnicalServiceId>
  <dep:T2SMessageId>T2S.MSGRT.NSPname1.20110101000000.000003</dep:T2SMessageId>
  <dep:SendTimestamp>2011-01-01T00:00:00</dep:SendTimestamp>
  <dep:DeliveryMode>RT</dep:DeliveryMode>
  <dep:ExchangeStatus>OK</dep:ExchangeStatus>
 </dep:ExchangeHeader>
 <dep:BusinessEnvelope>
  <dep:BusinessApplicationHeader>
   <!-- business application header goes here -->
  </dep:BusinessApplicationHeader>
  <dep:BusinessMessage>
   <!-- business message goes here -->
  </dep:BusinessMessage>
 </dep:BusinessEnvelope>
</dep:Request>
```

Technical Ack (data part) sent by the NSP's gateway or DCP to the T2S Platform at the step no. 3

```
<?xml version="1.0" encoding="UTF-8" ?>
<dep:TechnicalAck
 xmlns:dep="http://www.ecb.eu/t2s-0.2"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.ecb.eu/t2s-0.2 dep-02.xsd ">
 <dep:ExchangeHeader>
  <dep:Version>0.2</dep:Version>
  <dep:Sender>cn=t2sappl1,o=t2sprod</dep:Sender>
  <dep:Receiver>cn=t2s-cust1,o=nsp-name1</dep:Receiver>
  <dep:TechnicalServiceId>nsp-name1.MSGRT.PROD</dep:TechnicalServiceId>
  <dep:NSPCommunicationId>nsp-name1.gtw134567.20100908185555.123456</dep:NSPCommunicationId>
  <dep:T2SMessageId>T2S.MSGRT.NSPname1.20110101000000.000003</dep:T2SMessageId>
  <dep:EntryTimestamp>2011-01-01T00:00:00</dep:EntryTimestamp>
  <dep:SendTimestamp>2011-01-01T00:00:01</dep:SendTimestamp>
  <dep:DeliveryMode>RT</dep:DeliveryMode>
  <dep:ExchangeStatus>OK</dep:ExchangeStatus>
 </dep:ExchangeHeader>
</dep:TechnicalAck>
```

Response sent by NSP's gateway or DCP to T2S Platform at the step no. 6

```
<?xml version="1.0" encoding="UTF-8" ?>
<dep:Response
 xmlns:dep="http://www.ecb.eu/t2s-0.2"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.ecb.eu/t2s-0.2 dep-02.xsd ">
 <dep:ExchangeHeader>
  <dep:Version>0.2</dep:Version>
  <dep:Sender>cn=t2s-cust1,o=nsp-name1</dep:Sender>
  <dep:Receiver>cn=t2sappl1,o=t2sprod</dep:Receiver>
  <dep:TechnicalServiceId>nsp-name1.MSGRT.PROD</dep:TechnicalServiceId>
  <dep:NSPCommunicationId> nsp-name1.gtw134567.20100908185555.123456</dep:NSPCommunicationId>
  <dep:T2SActorMessageId>T2SActorGateway1.20100908175531.123456</dep:T2SActorMessageId>
```

```
  <dep:EntryTimestamp>2011-01-01T00:00:00</dep:EntryTimestamp>
  <dep:SendTimestamp>2011-01-01T00:00:00</dep:SendTimestamp>
  <dep:DeliveryMode>RT</dep:DeliveryMode>
  <dep:ExchangeStatus>OK</dep:ExchangeStatus>
 </dep:ExchangeHeader>
 <dep:BusinessEnvelope>
  <dep:BusinessApplicationHeader>
   <!-- business application header goes here -->
  </dep:BusinessApplicationHeader>
  <dep:BusinessMessage>
   <!-- business message goes here -->
  </dep:BusinessMessage>
 </dep:BusinessEnvelope>
</dep:Response>
```

Message sent by T2S Platform to the NSP's gateway or DCP at the step no. 2:

```
<?xml version="1.0" encoding="UTF-8" ?>
<dep:Request
 xmlns:dep="http://www.ecb.eu/t2s-0.2"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.ecb.eu/t2s-0.2 dep-02.xsd ">
 <dep:ExchangeHeader>
  <dep:Version>0.2</dep:Version>
  <dep:Sender>cn=t2sappl1,o=t2sprod</dep:Sender>
  <dep:Receiver>cn=t2s-cust1,o=nsp-name1</dep:Receiver>
  <dep:TechnicalServiceId>nsp-name1.MSGRT.PROD</dep:TechnicalServiceId>
  <dep:T2SMessageId>T2S.MSGRT.NSPname1.20110101000000.000003</dep:T2SMessageId>
  <dep:SendTimestamp>2011-01-01T00:00:00</dep:SendTimestamp>
  <dep:DeliveryMode>RT</dep:DeliveryMode>
  <dep:ExchangeStatus>OK</dep:ExchangeStatus>
 </dep:ExchangeHeader>
 <dep:BusinessEnvelope>
  <dep:BusinessApplicationHeader>
   <!-- business application header goes here -->
  </dep:BusinessApplicationHeader>
  <dep:BusinessMessage>
   <!-- business message goes here -->
  </dep:BusinessMessage>
 </dep:BusinessEnvelope>
</dep:Request>
```

Technical Ack (data part) sent by the NSP's gateway or DCP  to the T2S Platform at the step no. 3

```
<?xml version="1.0" encoding="UTF-8" ?>
<dep:TechnicalAck
 xmlns:dep="http://www.ecb.eu/t2s-0.2"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.ecb.eu/t2s-0.2 dep-02.xsd ">
 <dep:ExchangeHeader>
  <dep:Version>0.2</dep:Version>
  <dep:Sender>cn=t2sappl1,o=t2sprod</dep:Sender>
  <dep:Receiver>cn=t2s-cust1,o=nsp-name1</dep:Receiver>
  <dep:TechnicalServiceId>nsp-name1.MSGRT.PROD</dep:TechnicalServiceId>
  <dep:NSPCommunicationId>nsp-name1.gtw134567.20100908185555.123456</dep:NSPCommunicationId>
  <dep:T2SMessageId>T2S.MSGRT.NSPname1.20110101000000.000003</dep:T2SMessageId>
  <dep:EntryTimestamp>2011-01-01T00:00:00</dep:EntryTimestamp>
  <dep:SendTimestamp>2011-01-01T00:00:01</dep:SendTimestamp>
  <dep:DeliveryMode>RT</dep:DeliveryMode>
  <dep:ExchangeStatus>OK</dep:ExchangeStatus>
 </dep:ExchangeHeader>
</dep:TechnicalAck>
```

Response sent by NSP's gateway or DCP to T2S Platform at the step no. 6

```
<?xml version="1.0" encoding="UTF-8" ?>
<dep:Response
 xmlns:dep="http://www.ecb.eu/t2s-0.2"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.ecb.eu/t2s-0.2 dep-02.xsd ">
 <dep:ExchangeHeader>
  <dep:Version>0.2</dep:Version>
  <dep:Sender>cn=t2s-cust1,o=nsp-name1</dep:Sender>
  <dep:Receiver>cn=t2sappl1,o=t2sprod</dep:Receiver>
  <dep:TechnicalServiceId>nsp-name1.MSGRT.PROD</dep:TechnicalServiceId>
```

```
  <dep:NSPCommunicationId> nsp-name1.gtw134567.20100908185555.123456</dep:NSPCommunicationId>
  <dep:T2SActorMessageId>T2SActorGateway1.20100908175531.123456</dep:T2SActorMessageId>
  <dep:EntryTimestamp>2011-01-01T00:00:00</dep:EntryTimestamp>
  <dep:SendTimestamp>2011-01-01T00:00:00</dep:SendTimestamp>
  <dep:DeliveryMode>RT</dep:DeliveryMode>
  <dep:ExchangeStatus>OK</dep:ExchangeStatus>
 </dep:ExchangeHeader>
 <dep:BusinessEnvelope>
  <dep:BusinessApplicationHeader>
   <!-- business application header goes here -->
  </dep:BusinessApplicationHeader>
  <dep:BusinessMessage>
   <!-- business message goes here -->
  </dep:BusinessMessage>
 </dep:BusinessEnvelope>
</dep:Response>
```

"Request" message sent by the T2S Platform at step no.2

```
<?xml version="1.0" encoding="UTF-8" ?>
<dep:Request
 xmlns:dep="http://www.ecb.eu/t2s-0.2"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.ecb.eu/t2s-0.2 dep-02.xsd ">
 <dep:ExchangeHeader>
  <dep:Version>0.2</dep:Version>
  <dep:Sender>cn=t2sappl1,o=t2sprod</dep:Sender>
  <dep:Receiver>cn=t2s-cust1,o=nsp-name1</dep:Receiver>
  <dep:TechnicalServiceId>nsp-name1.MSGSNF.PROD</dep:TechnicalServiceId>
  <dep:T2SMessageId>T2S.MSGSNF.NSPname1.20110101000000.000005</dep:T2SMessageId>
  <dep:SendTimestamp>2011-01-01T00:04:01</dep:SendTimestamp>
  <dep:DeliveryMode>SF</dep:DeliveryMode>
  <dep:ExchangeStatus>OK</dep:ExchangeStatus>
 </dep:ExchangeHeader>
 <dep:BusinessEnvelope>
  <dep:BusinessApplicationHeader>
   <!-- business application header goes here -->
  </dep:BusinessApplicationHeader>
  <dep:BusinessMessage>
   <!-- business message goes here -->
  </dep:BusinessMessage>
 </dep:BusinessEnvelope>
</dep:Request>
```

Data part of the Technical Ack received by the T2S Platform at step no. 3

```
<?xml version="1.0" encoding="UTF-8" ?>
<dep:TechnicalAck
 xmlns:dep="http://www.ecb.eu/t2s-0.2"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.ecb.eu/t2s-0.2 dep-02.xsd ">
 <dep:ExchangeHeader>
  <dep:Version>0.2</dep:Version>
  <dep:Sender>cn=t2sappl1,o=t2sprod</dep:Sender>
  <dep:Receiver>cn=t2s-cust1,o=nsp-name1</dep:Receiver>
  <dep:TechnicalServiceId>nsp-name1.MSGSNF.PROD</dep:TechnicalServiceId>
  <dep:NSPCommunicationId>nsp-name1.gtw134567.20100908185555.123456</dep:NSPCommunicationId>
  <dep:T2SMessageId>T2S.MSGSNF.NSPname1.20110101000000.000005</dep:T2SMessageId>
  <dep:SendTimestamp>2011-01-01T00:04:01</dep:SendTimestamp>
  <dep:DeliveryMode>SF</dep:DeliveryMode>
  <dep:ExchangeStatus>OK</dep:ExchangeStatus>
 </dep:ExchangeHeader>
</dep:TechnicalAck>
```

DeliveryNotif failure message received by the T2S Platform in the case of a delivery notification failure from the NSP's gateway or DCP because the final receiver has not connected for 14 calendar days

```
<?xml version="1.0" encoding="UTF-8" ?>
<dep:DeliveryNotification
 xmlns:dep="http://www.ecb.eu/t2s-0.2"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.ecb.eu/t2s-0.2 dep-02.xsd ">
 <dep:ExchangeHeader>
  <dep:Version>0.2</dep:Version>
```

```
  <dep:Sender>cn=t2sappl1,o=t2sprod</dep:Sender>
  <dep:Receiver>cn=t2s-cust1,o=nsp-name1</dep:Receiver>
  <dep:TechnicalServiceId>nsp-name1.MSGSNF.PROD</dep:TechnicalServiceId>
  <dep:NSPCommunicationId>nsp-name1.gtw134567.20100908185555.123456</dep:NSPCommunicationId>
  <dep:T2SMessageId>T2S.MSGSNF.NSPname1.20110101000000.000003</dep:T2SMessageId>
  <dep:EntryTimestamp>2011-01-01T00:00:00</dep:EntryTimestamp>
  <dep:SendTimestamp>2011-01-01T00:00:01</dep:SendTimestamp>
  <dep:DeliveryMode>SF</dep:DeliveryMode>
  <dep:ExchangeStatus>KO</dep:ExchangeStatus>
  <dep:ErrorDescription>
    <dep:ErrorCode>T2S040E</dep:ErrorCode>
    <dep:AdditionalInfo>
              Message expired. Receiver has not been connected for 14 days
    </dep:AdditionalInfo>
  </dep:ErrorDescription>
 </dep:ExchangeHeader>
</dep:DeliveryNotification>
```

Example of message to enable/disable the SnF traffic

```
<?xml version="1.0" encoding="UTF-8" ?>
<dep:EnableSnfTraffic
 xmlns:dep="http://www.ecb.eu/t2s-0.2"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.ecb.eu/t2s-0.2 dep-02.xsd ">
       <dep:Service>
              <dep:Name>nsp-name1.MSGSF.PROD</dep:Name>
              <dep:DestQmanagerName>WQI1</dep:DestQmanagerName>
              <dep:DestQueueName>IGN.NSPNAME1.MSGSF.INCOMING.L01</dep:DestQueueName>
       </dep:Service>
       <dep:Service>
              <dep:Name>nsp-name1.FILESF.PROD</dep:Name>
              <dep:DestQmanagerName>WQI1</dep:DestQmanagerName>
              <dep:DestQueueName>IGN.NSPNAME1.FILESF.INCOMING.L01</dep:DestQueueName>
       </dep:Service>
</dep:EnableSnfTraffic>
```

An example of the response of the NSP's gateway or DCP  to this message is the following

```
<?xml version="1.0" encoding="UTF-8" ?>
<dep:EnableSnfTrafficAck
 xmlns:dep="http://www.ecb.eu/t2s-0.2"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.ecb.eu/t2s-0.2 dep-02.xsd ">
 <dep:Service>
  <dep:Name>nsp-name1.MSGSF.PROD</dep:Name>
  <dep:DestQmanagerName>WQI1</dep:DestQmanagerName>
  <dep:DestQueueName>IGN.NSPNAME1.MSGSF.INCOMING.L01</dep:DestQueueName>
  <dep:Status>Activated</dep:Status>
  <dep:Reason/>
 </dep:Service>
 <dep:Service>
  <dep:Name>nsp-name1.FILESF.PROD</dep:Name>
  <dep:DestQmanagerName>WQI1</dep:DestQmanagerName>
  <dep:DestQueueName>IGN.NSPNAME1.FILESF.INCOMING.L01</dep:DestQueueName>
  <dep:Status>Failed</dep:Status>
  <dep:Reason>Queue not accessible. MQRC=2035</dep:Reason>
 </dep:Service>
</dep:EnableSnfTrafficAck>
```