ETSI TS 119 534-2 V1.1.1 (2019-02)

**Electronic Signatures and Infrastructures (ESI);
Testing Conformance and Interoperability of
Registered Electronic Mail Services;
Part 2: Test suites for interoperability testing of
providers using same format and transport protocols**

Reference

DTS/ESI-0019534-2

Keywords

interoperability, registered electronic mail, testing

Contents

Intellectual Property Rights

Essential patents

Trademarks

Foreword

This Technical Specification (TS) has been produced by ETSI Technical Committee Electronic Signatures and Infrastructures (ESI).

The present document is part 2 of a multi-part deliverable. Full details of the entire series can be found in part 1 [i.1].

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

1 Scope

The present document defines:

1) A test suite for supporting interoperability tests within the field of Registered Electronic Mail (REM hereinafter) as specified in ETSI EN 319 532 parts 1 [3], 2 [4], 3 [5] and 4 [6]. The test suite defines test cases for the following environments:

   - Environments that correspond to the basic model as defined in ETSI EN 319 532-1 [3] where sender and all the entities at receiving side are subscribed to the same REMS. Test cases are defined for REMSs operating Store&Forward and for REMSs operating Store&Notify styles.

   - Environments that correspond to the 4-corner model as defined in ETSI EN 319 532-1 [3] where sender is subscribed to one REMS and the entities at receiving side are subscribed to another one, and no intermediate REMS is required for relaying REM messages between them. Test cases are defined for covering the three possible different combinations of styles, namely Store&Forward to Store&Forward, Store&Forward to Store&Notify, and Store&Notify to Store&Forward.

   - Environments that correspond to the extended model as defined in ETSI EN 319 532-1 [3] where sender is subscribed to one REMS and the entities at receiving side are subscribed to another one, and intermediate REMSs are required for relaying REM messages between them. Test cases are defined for covering two different combinations of styles, namely Store&Forward to Store&Forward to Store&Forward, Store&Forward to Store&Notify to Store&Forward.

2) A mechanism for documenting new test cases and expanding the aforementioned test suite.

2 References

2.1 Normative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found at https://docbox.etsi.org/Reference.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are necessary for the application of the present document.

[1] ETSI EN 319 522-1: "Electronic Signatures and Infrastructures (ESI); Electronic Registered Delivery Services; Part 1: Framework and Architecture".

[2] ETSI EN 319 522-2: "Electronic Signatures and Infrastructures (ESI); Electronic Registered Delivery Services; Part 2: Semantic contents".

[3] ETSI EN 319 532-1: "Electronic Signatures and Infrastructures (ESI); Registered Electronic Mail (REM) Services; Part 1: Framework and Architecture".

[4] ETSI EN 319 532-2: "Electronic Signatures and Infrastructures (ESI); Registered Electronic Mail (REM) Services; Part 2: Semantic Contents".

[5] ETSI EN 319 532-3: "Electronic Signatures and Infrastructures (ESI); Registered Electronic Mail (REM) Services; Part 3: Formats".

[6] ETSI EN 319 532-4: "Electronic Signatures and Infrastructures (ESI); Registered Electronic Mail (REM) Services; Part 4: Interoperability profiles".

[7] IETF RFC 8118: "The application/pdf Media Type".

[8] IETF RFC 2046: "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types".

[9] IETF RFC 2183: " Communicating Presentation Information in Internet Messages: The Content-Disposition Header Field".

[10] IETF RFC 5751: "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification".

[11] IETF RFC 5322: "Internet Message Format".

[12] IETF RFC 2854: "The 'text/html' Media Type".

[13] IETF RFC 7303: "XML Media Types".

[14] IETF RFC 2045: "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies".

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] ETSI TS 119 534-1: "Electronic Signatures and Infrastructures (ESI); Testing Conformance and Interoperability of Registered Electronic Mail Services; Part 1: Testing conformance".

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the terms given in ETSI EN 319 532-1 [3] apply.

3.2 Symbols

Void.

3.3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

ACC_REJ_EXP	ACCeptanceREJectionEXPiry
CONS_ACC	CONSignmentACCeptance
CONS_NOT	CONSignmentNOTification
CONS_NOT_FAIL	CONSignmentNOTificationFAILure
CONS_REJ	CONSignmentREJection
CONT_CONS	CONTentCONSignment
CONT_CONS_FAIL	CONTentCONSignmentFAILure
CONT_HAND	CONTentHANDover
CONT_HAND_FAIL	CONTentHANDoverFAILure
ERDS	Electronic Registered Delivery
EV_SET	EVidence SET
IREMS	Intermediate Registered Electronic Mail Service

NOT_F_ACC	NOTificationForACCeptance
NOT_F_ACC_FAIL	NOTificationForACCeptanceFAILure
REC_F_NERDS	RECeivedFromNonERDS
REL_ACC	RELayACCeptance
REL_FAIL	RELayFAILure
REL_REJ	RELayREJection
REL_T_NERDS	RELayToNonERDS
REL_T_NERDS_FAIL	RELayToNonERDSFAILure
REM	Registered Electronic Mail
REMS	Registered Electronic Mail Service
RREMS	Recipient's Registered Electronic Mail Service
SCN_ID	Scenario Identifier
SMIME	Secure/Multipurpose Internet Mail Extensions
SREMS	Sender's Registered Electronic Mail Service
SUB_ACC	SUBmissionACCeptance
SUB_REJ	SUBmissionREJection

4 Technical approach

4.1 Components of test cases and their identifiers

As it has been mentioned before the present document defines:

1) A test suite for supporting interoperability tests within the field of Registered Electronic Mail (REM hereinafter) as specified in ETSI EN 319 532 parts 1 [3], 2 [4], 3 [5] and 4 [6].

2) A mechanism for documenting new test cases and expanding the aforementioned test suite.

The present document follows a layered approach for building the definition of the test cases in the test suite, which can be summarized as follows:

1) Clause 5 defines a number of parameterized scenarios. A scenario consists of a number of entities, namely: sender, one or more REMSs, and the entities at receiving side - one or more recipients and/or one or more recipients' delegates -, which exchange different REM messages with time. Each scenario corresponds to one of the three models presented in ETSI EN 319 532-1 [3]. This clause presents a template for defining one scenario, in a way that resembles to some templates used for defining use cases scenarios in software engineering.

   This template:

   - Includes the enumeration of all the REM messages exchanged by the participating entities. This list of exchanged REM messages is one of the parameters of the scenario.

   - Also includes a list of ERDS evidence sets, which, in the scenario, are incorporated in some REM messages.

   One scenario may be used for defining several test cases depending on:

   - The specific components of each exchanged REM message (suppressing or adding an optional header, or changing the value of a certain header field results in a different REM message and consequently a different test case).

   - The entities at receiving part (for instance, changing one recipient by one recipient's delegate, or two recipients and one recipient's delegate results in a different the test case).

   - A named set of additional requirements (for instance whether the original message contains or not attachments, is signed, is encrypted, etc.).

   - In negative test cases, i.e. test cases where the service failed in consigning or handing over the message to one or more recipients, the reason(s) causing that failure.

This means that one test case corresponds to one scenario where all the exchanged REM messages have been completely defined in terms of their components, all the participating entities have been established, and all the additional requirements have also been defined. Taking the functional notation this can be expressed as follows:

> **TestCase#i = Scenario_id(<Receiving side identifier>, <REM message identifier 1>, <REM message identifier 2>, …, <REM message identifier N>, <additional requirements set identifier>,<failure reasons>?**

Where:

- <Receiving side identifier> is the identifier assigned to a certain set of entities at receiving side;
- <REM message identifier I> is the identifier of a specific instantiation of the aforementioned REM message, namely: REM payload, REM notification, REM Receipt, or REM dispatch, which are defined in clauses 6.3, 6.4, 6.5 and 6.6 respectively.
- <additional requirements set identifier> is the identifier of a named set of additional requirements. Clause 7.2 defines a number of these named sets.
- <failure reason(s)>? is the reason(s) that caused that the service failed in consigning or handing over the message to the recipient(s). It shall only appear in negative test cases.

2) Clauses 6.3, 6.4, 6.5 and 6.6 define specific instantiations of REM payloads, REM notifications, REM receipts and REM dispatches respectively. Each type of REM message is composed by several MIME sections, with their headers and bodies. One specific instantiation of a certain type of REM message is composed by one specific combination of MIME sections. Each MIME section in turn is formed by one certain combination of headers and has a specific value in its body. The present document defines a number of combinations of MIME sections in clauses 6.2.2, 6.2.3, 6.2.4.3, 6.2.4.4, 6.2.5, 6.2.6, 6.2.7 and 6.2.8, and assigns to each one a unique identifier. This allows to use again the functional notation, and define one instantiation of a certain type of REM message as follows:

   REM message instance = Sequence(<outer most MIME header identifier>, <signed data MIME header section identifier>, <multipart/alternative free text MIME section identifier>, <multipart/alternative html MIME section>, <original message MIME section identifier>?, <extension MIME section identifier>*, <ERDS evidence MIME section identifier>*,<signature MIME section identifier>)

Where ? indicates a cardinality 0 or 1 for the affected MIME section, and * indicates a cardinality of 0 or more for the affected MIME sections.

3) Clauses 6.2.2, 6.2.3, 6.2.4.3, 6.2.4.4, 6.2.5, 6.2.6, 6.2.7 and 6.2.8 define specific instances for the outermost MIME header, the signed data MIME header, the multipart/alternative free text MIME section, the multipart/alternative html MIME section, original message MIME section, the extension MIME section, the ERDS evidence MIME section, and the signature MIME section respectively. Each clause define different instances of the aforementioned headers and sections and assigns them unique identifiers that are used for defining specific instances of the different REM messages as shown above. Once this level is reached, the specific test case is fully defined as: a scenario where fully defined, REM messages are exchanged between a specific set participating entities, and where a specific set of additional requirements are imposed.

4.2 Adding new test cases to the test suite

The strategy followed for building the definitions of the test cases makes it easy to expand the test suite by incorporation of new test cases.

For defining a new test case the following steps are required:

1) Identify the **set of receiving entities**. If none of the predefined set of entities at the receiving side is the one required, define a new set as specified in clause 7.3. The sender is always present by default.
2) Define the REMSs that will participate in the test case.
3) If the set of participating REMSs is not equal to none of the scenarios already identified in the present document, the new scenario will require to be defined in a new template.

4) Identify the **sequence of actions** performed by each actor and their order of occurrence and assign a new unique identifier (**<SCN_ID >)** to the scenario.

5) Identify **all the REM messages** generated by the actors as they go through the sequence of actions. For each message:

   a) Identify its MIME sections.

   b) For each MIME section identified different than the ERDS MIME sections, check if its header fields combination and the corresponding bodies have already been defined in the present document. If not, add the required combination of header fields and body values to the repertoire of named combinations to the section defining instances of the aforementioned MIME section as in the corresponding clauses (clauses 6.2.2, 6.2.3, 6.2.4.3, 6.2.4.4, 6.2.6 or 6.2.8).

   c) List all the REM messages exchanged as parameters of the scenario.

   d) Identify the ERDS evidence format and the set of ERDS evidence for each REM message including them and add the names of the ERDS evidence sets to the Var section of the template.

6) Identify and define any other additional requirement for completely define the test case. If the set of requirements is different than all the sets already define, assign a name to it (**<ADD_REQ_COMB>)** and add it to the repertoire of named sets of additional requirements in Table 23 (clause 7.2).

5 Scenarios

5.1 Introduction

The present clause defines a number of selected scenarios that will be used in clause 8.

Clause 5.3 defines scenarios where sender and recipient(s) are subscribed to the same REMS.

Clause 5.4 defines scenarios where the sender and the recipient(s) are subscribed to different REMSs and there are not intermediate REMSs between them.

Clause 5.5 defines scenarios where sender is subscribed to a REMS and the recipient(s) is(are) not subscribed to the same REMS and there are one or more intermediate REMSs.

Unless anything said against it, all the scenarios assume that there are N entities at the receiving side.

Unless anything said against it, all the ERDS evidences that can contain details of different entities at the receiving side shall incorporate the details of the entire set of N entities at the receiving side.

Table 1 shows the template for defining one scenario.

Table 1: Template for the tabular definition of one scenario

Scenario id: <SCN_ID>				**Purpose**
Parameter: <REMS_receipt>_with_XML_SUB_REJ *<Parameter 1 that helps to fully specify the scenario. Their number depends on the specific scenario>*		Var EV_SET#1 = {…, …} *Named sets of ERDS evidence used in the definition of the scenario.*		
Parameter: *<Parameter 2>*		Var EV_SET#2 = {… …}		
Parameter: *<Parameter N>*		Var EV_SET#N = {… …}		
Sequence of actions				
<SEQUENCE OF ACTIONS. THERE IS ONE COLUMN PER PARTICIPATING ACTOR>				
#	**Sender**	**REMS**	**Receiving side**	
The sequence is composed of a number of numerated steps, when the actors perform certain actions as shown below. Some frequent actions: send original message, accept submission, reject submission, consign, generate one ERDS evidence, generate one REM message, etc.				
1	Sender sends original message			
2		Rejects submission. Generates XML_SUB_REJ ERDS evidence		
3		Generates <REMS_receipt>_with_XML_SUB_REJ		
4		Sends <REMS_receipt>_with_XML_SUB_REJ		
5	Receives <REMS_receipt>_with_XML_SUB_REJ			

Each scenario is assigned a unique identifier <SCN_ID>. The reasons why the scenario has been defined are shown in column "Purpose".

The definition of each scenario requires that parties exchange a number of REM messages, which appear listed as parameters in the rows immediately below the headers row. Its number depends on the specific scenario.

Below the list of parameters the table shows a sequence of actions performed by different involved entities, which results in that a set of REM messages is generated and exchanged.

The definition of each scenario also can use a number of named ERDS evidence sets, which are listed in cells started with Var. Each ERDS evidence set is given a name EV_SET#<i>, being <i> a non-negative integer number.

The involved entities are sender (or sender's delegate, the scenario definition does not make any distinction between them), one or more REMSs, and the entities at the receiving side (for the same scenario there may be different sets of entities, for instance one recipient, one recipient's delegate, one or more recipients, or one or more recipients and one or more recipients' delegates).

Each step is assigned a positive integer number. The actions performed include: submission of messages, generation of REM messages, generation of ERDS evidence, acceptance of REM messages, rejection of REM messages, consignment of REM messages, retrieval of REM messages by entities at receiving side, failures, etc.

5.2 Abbreviations used in tables defining scenarios

The present clause shows some abbreviations (which have already been anticipated in clause 3.3 which have already been anticipated in clause 3.3) used in the tables that define the scenarios.

Table 2 shows the abbreviations used for the different ERDS evidence.

Table 2: ERDS evidence abbreviations

ERDS Evidence name	ERDS Evidence abbreviation
SubmissionAcceptance	SUB_ACC
SubmissionRejection	SUB_REJ
RelayAcceptance	REL_ACC
RelayRejection	REL_REJ
RelayFailure	REL_FAIL
NotificationForAcceptance	NOT_F_ACC
NotificationForAcceptanceFailure	NOT_F_ACC_FAIL
ConsignmentAcceptance	CONS_ACC
ConsignmentRejection	CONS_REJ
AcceptanceRejectionExpiry	ACC_REJ_EXP
ContentConsignment	CONT_CONS
ContentConsignmentFailure	CONT_CONS_FAIL
ConsignmentNotification	CONS_NOT
ConsignmentNotificationFailure	CONS_NOT_FAIL
ContentHandover	CONT_HAND
ContentHandoverFailure	CONT_HAND_FAIL
RelayToNonERDS	REL_T_NERDS
RelayToNonERDSFailure	REL_T_NERDS_FAIL
ReceivedFromNonERDS	REC_F_NERDS

ETSI EN 319 522-1 [1] specify a XML format for ERDS evidence, but also allows that they are PDF documents. The present document differentiates both cases using a prefix for the ERDS evidence abbreviations as follows:

- **XML_** prefix indicates that the identified object is a XML ERDS evidence;
- **PDF_** prefix that the identified object is a PDF ERDS evidence.

EXAMPLE: The abbreviation for the XML SubmissionAcceptance ERDS evidence will be **XML_SUB_ACC**. The abbreviation for the PDF SubmissionAcceptance ERDS evidence will be **PDF_SUB_ACC.**

The tables defining the Scenarios use the following abbreviations for the different participating REMSs:

- **SREMS** stands for the REMS serving the sender, in the scenarios where it is different from the REMS serving the entities at receiving side.
- **RREMS** stands for the REMS serving the entities at receiving side, in the scenarios where it is different from the REMS serving the sender.
- **IREMS** stands for a REMS that directly serves neither the sender nor the recipient(s)/recipient's delegate, but instead is an intermediate REMS that relays REM messages from SREMS to RREMS and from RREMS to SREMS.

5.3 Black-box model scenarios

5.3.1 Introduction

This clause defines scenarios where the sender and the entities at the receiving side are subscribed to the same REMS and consequently REM messages are not relayed between different REMSs.

Clause 5.3.2 defines scenarios where the REMS operates in Store and Forward style.

Clause 5.3.3 defines scenarios where the REMS operates in Store and Notify style.

5.3.2 Scenarios for Store and Forward style

Table 3 defines a number of scenarios for the case where sender and the entities at receiving side are subscribed to the same REMS operating in Store and Forward style.

Table 3: Scenarios for intra-REMS operating in Store and Forward style (1/13)

Scenario id: REMS_SF#1				Purpose
Parameter: <REMS_receipt>_with_XML_SUB_REJ				The simplest scenario: the REMS rejects the original message submitted by the sender because a unique reason, and sends back a REM receipt with the SubmissionRejection ERDS evidence
Sequence of actions				
#	**Sender**	**REMS**	**Receiving side**	
1	Sender sends original message			
2		Rejects submission. Generates SUB_REJ ERDS evidence with details of the N recipients		
3		Generates <REMS_receipt>_with_XML_SUB_REJ		
4		Sends REMS receipt to the sender		
5	Receives REMS receipt			

NOTE 1: As it has been anticipated, negative scenarios like this one do not mention the reason for failure. This is a separated parameter for the test case definition in clause 8.

Table 3a: Scenarios for intra-REMS operating in Store and Forward style (2/13)

Scenario id: REMS_SF#2				Purpose
Parameter: <REMS_receipt>_with_XML_SUB_REJ		Var EV_SET#1 = {2 XML_SUB_REJ }		As before but now the REMS rejects the original message submitted by the sender because of one reason for M entities at the receiving side and because another reason for the other N-M entities. It generates two SubmissionRejection ERDS evidences it and sends back a REM receipt with these two SubmissionRejection ERDS evidences
Sequence of actions				
#	**Sender**	**REMS**	**Receiving side**	
1	Sender sends original message			
2		Rejects submission. Generates 2 XML_SUB_REJ ERDS evidences. One of them with details of M entities; the other with details of N-M entities		
3		Generates <REMS_receipt>_with the 2_aforementioned XML_SUB_REJ		
4		Sends REMS receipt to the sender		
5	Receives REMS receipt			

Table 3b: Scenarios for intra-REMS operating in Store and Forward style (3/13)

Scenario id: REMS_SF#3				**Purpose**
Parameter: <REM_dispatch>_with_XML_SUB_ACC		Var EV_SET#1 = {XML_SUB_ACC , XML_CONT_CONS}		The simplest successful scenario: the REMS accepts the submission of the original message, generates as many REM dispatches as required by the number of entities in the receiving side enclosing the original message and the SubmissionAcceptance ERDS evidence, and delivers them to the receiving side. After that it generates and sends back to the sender a REM receipt with one SubmissingAcceptance ERDS evidence and one ContentConsignment ERDS evidence. **Each evidence includes the details of all the (N) entities at receiving side**
Parameter: <REMS_receipt>_with_EV_SET#1				
Sequence of actions				
#	**Sender**	**REMS**	**Receiving side**	
1	Sender sends original message			
2		Accepts submission. Generates XML_SUB_ACC ERDS evidence		
3		Generates <REM_dispatch>_with_XML_SUB_ACC for the N receiving entities		
4		Consigns REM dispatch to receiving side		
5		Generates one XML_CONT_CONS ERDS evidence with details of the N recipients	<REM_dispatch>_with_XML_SUB_ACC consigned to receiving side	
6		Generates <REMS_receipt>_with_EV_SET#1		
7		Sends it back to sender		
8	Receives <REMS_receipt>_with_EV_SET#1			

Table 3c: Scenarios for intra-REMS operating in Store and Forward style (4/13)

Scenario id: REMS_SF#4				Purpose
Parameter: <REM_dispatch>_with_XML_SUB_ACC		Var EV_SET#1 = {XML_SUB_ACC , XML_CONT_CONS, XML_CONT_CONS_FAIL}		As scenario REMS_SF#3 but now one consignment fails
Parameter: <REMS_receipt>_with_EV_SET#1				
Sequence of actions				
#	Sender	REMS	Receiving side	
1	Sender sends original message			
2		Accepts submission. Generates XML_SUB_ACC ERDS evidence		
3		Generates <REM_dispatch>_with_XML_SUB_ACC for the N receiving entities		
4		Consigns REM dispatch to receiving side		
5		Generates one XML_CONT_CONS ERDS evidence with details of the N-1 entities AND one XML_CONT_CONS_FAIL with details of one entity	N-1 <REM_dispatch>_with_XML_SUB_ACC consigned to receiving side 1 fails	
6		Generates <REMS_receipt>_with_EV_SET#1		
7		Sends it back to sender		
8	Receives <REMS_receipt>_with_EV_SET#1			

Table 3d: Scenarios for intra-REMS operating in Store and Forward style (5/13)

Scenario id: REMS_SF#5				**Purpose**
Parameter: <REM_dispatch>_with_XML_SUB_ACC		Var EV_SET#1 = {XML_SUB_ACC , XML_CONT_CONS, 2 XML_CONT_CONS_FAIL}		As scenario REMS_SF#3 but now two consignments fail, and each one for different reason, which implies the generation of two XML_CONT_CONS_FAIL ERDS evidences
Parameter: <REMS_receipt>_with_EV_SET#1				
Sequence of actions				
#	**Sender**	**REMS**	**Receiving side**	
1	Sender sends original message			
2		Accepts submission. Generates XML_SUB_ACC ERDS evidence		
3		Generates <REM_dispatch>_with_XML_SUB_ACC for the N receiving entities		
4		Consigns REM dispatch to receiving side		
5		Generates one XML_CONT_CONS ERDS evidence with details of the N-2 entities AND 2 XML_CONT_CONS_FAIL with details of one entity	N-2 <REM_dispatch>_with_XML_SUB_ACC consigned to receiving side The other 2 fail, each one for a different reason	
6		Generates <REMS_receipt>_with_EV_SET#1		
7		Sends it back to sender		
8	Receives <REMS_receipt>_with_EV_SET#1			

Table 3e: Scenarios for intra-REMS operating in Store and Forward style (6/13)

Scenario id: REMS_SF#6				Purpose
Parameter: <REM_dispatch>_with_XML_SUB_ACC		Var EV_SET#1 = {XML_SUB_ACC, XML_CONT_CONS, XML_CONT_HAND}		Successful scenario. After the successful consignment, all the entities at the receiving side successfully retrieve the messages. The REMS generates the following ERDS evidence: one SubmissiongAcceptance, one ContentConsignment, and one ContentHandover **Each ERDS evidence includes the details of all the (N) entities at the receiving side**
Parameter: <REMS_receipt>_with_EV_SET#1				
Sequence of actions				
#	**Sender**	**REMS**	**Receiving side**	
1	Sender sends original message			
2		Accepts submission. Generates XML_SUB_ACC ERDS evidence		
3		Generates <REM_dispatch>_with_XML_SUB_ACC		
4		Consigns them to the receiving side		
5		Generates one XML_CONT_CONS ERDS evidence with details of the N recipients	N <REM_dispatch>_with_XML_SUB_ACC consigned to receiving side	
6			All the entities retrieve the REM dispatch	
7		Generates one XML_CONT_HAND ERDS evidence with details of the N recipients	N <REM_dispatch>_with_XML_SUB_ACC handed over to receiving side	
8		Generates <REMS_receipt>_with_EV_SET#1		
9	Receives <REMS_receipt>_with_EV_SET#1			

NOTE 2: The former scenarios explicitly mention in the Sequence of Actions columns that the ERDS evidences include details of the N recipients. As it has been anticipated above, hereinafter the absence of such explicit mention will be interpreted as an explicit mention of this fact. Explicit details will appear wherever this is not true.

Table 3f: Scenarios for intra-REMS operating in Store and Forward style (7/13)

Scenario id: REMS_SF#7				**Purpose**
Parameter: <REM_dispatch>_with_XML_SUB_ACC		Var EV_SET#1 = {XML_SUB_ACC, XML_CONT_CONS, XML_CONT_HAND, XML_CONT_HAND_FAIL}		As scenario REMS_SF#3 but now the handover of the REM dispatch is shown in the scenario, and one of the handing over fails. Each evidence includes the details of the entire set of recipient entities affected by the event that it reports, namely: • XML_SUB_ACC includes details of the N entities at the receiving side; • XML_CONT_CONS includes details of the N entities at the receiving side; • XML_CONT_HAND includes details of the N-1 entities that the service handed over the message to; • XML_CONT_HAND_FAIL includes the details of the entity that the service could not hand over the message to.
Parameter: REMS_receipt>_with_EV_SET#1				
Sequence of actions				
#	**Sender**	**REMS**	**Receiving side**	
1	Sender sends original message			
2		Accepts submission. Generates one XML_SUB_ACC ERDS evidence		
3		Generates <REM_dispatch>_with_XML_SUB_ACC		
4		Consigns them to the receiving side		
5		Generates one XML_CONT_CONS ERDS evidence	<REM_dispatch>_with_XML_SUB_ACC consigned to receiving side	
6			Entities try to retrieve the REM dispatch, but one fails	
7		Generates one XML_CONT_HAND ERDS one evicence for N-1 entities and one XML_CONT_HAND_FAIL ERDS evidence for one entity	N-1 <REM_dispatch>_with_XML_SUB_ACC handed over to receiving side. One fails	
8		Generates <REMS_receipt>_with_EV_SET#1		
9	Receives <REMS_receipt>_with_EV_SET#1			

Table 3g: Scenarios for intra-REMS operating in Store and Forward style (8/13)

Scenario id: REMS_SF#8				**Purpose**
Parameter: <REM_dispatch>_with_XML_SUB_ACC		Var EV_SET#1 = {XML_SUB_ACC, XML_CONT_CONS, XML_CONT_HAND, 2 XML_CONT_HAND_FAIL}		As scenario REMS_SF#3 but now the handover of the REM dispatch is shown in the scenario, but two of the handing overs fail Each evidence includes the details of the entire set of recipient entities affected by the event that it reports, namely: • XML_SUB_ACC includes details of the N entities at the receiving side; • XML_CONT_CONS includes details of the N entities at the receiving side; • tXML_CONT_HAND includes details of the N-2 entities that the service handed over the message to; • Each one of the two XML_CONT_HAND_FAIL ERDS evidence includes the details of the entity that the service could not hand over the message to with the reasons for not handing over. Hereinafter, the scenarios do not show handing over, but only consignment. However, a set of scenarios including handing over could be easily built based on them
Parameter: REMS_receipt>_with_EV_SET#1				
Sequence of actions				
#	**Sender**	**REMS**	**Receiving side**	
1	Sender sends original message			
2		Accepts submission. Generates one XML_SUB_ACC ERDS evidence		
3		Generates <REM_dispatch>_with_XML_SUB_ACC		
4		Consigns them to the receiving side		
5		Generates one XML_CONT_CONS ERDS evidence	<REM_dispatch>_with_XML_SUB_ACC consigned to receiving side	
6			Entities try to retrieve the REM dispatch, but one fails	
7		Generates one XML_CONT_HAND ERDS one evicence for N-2 entities and 2 XML_CONT_HAND_FAIL ERDS evidences, each one for a different entity	N-2 <REM_dispatch>_with_XML_SUB_ACC handed over to receiving side. Two of them fail	
8		Generates <REMS_receipt>_with_EV_SET#1		
9	Receives <REMS_receipt>_with_EV_SET#1			

Table 3h: Scenarios for intra-REMS operating in Store and Forward style (9/13)

Scenario id: REMS_SF#9				Purpose
Parameter: <REM_dispatch>		Var EV_SET#1 = {XML_SUB_ACC,XML_CONS_NOT, XML_CONT_CONS}		As scenario REMS_SF#3 but now the REMS sends a REMS notification of consignment to receiving side
Parameter: <REMS_notification>_of_Consignment				
Parameter: <REMS_receipt>_with_EV_SET#1				
Sequence of actions				
#	**Sender**	**REMS**	**Receiving side**	
1	Sender sends the original message			
2		Accepts submission. Generates one XML_SUB_ACC ERDS evidence		
3		Generates <REM_dispatch>_with_XML_SUB_ACC		
4		Consigns <REM_dispatch>_with_XML_SUB_ACC to receiving side		
5			N <REM_dispatch>_with_XML_SUB_ACC consigned to receiving side	
6		Generates one XML_CONT_CONS ERDS evidence		
7		Generates <REMS_notification>_of_Consignment for N entities		
8		Sends consignment notification to receiving side		
9		Generates one XML_CONS_NOT ERDS evidence for N entities	<REMS_notification>_of_Consignment received by receiving side	
10		Generates <REMS_receipt>_with_EV_SET#1		
11		Sends it to the sender		
12	Receives <REMS_receipt>_with_XML_XML_EV_SET #1			

Table 3i: Scenarios for intra-REMS operating in Store and Forward style (10/13)

Scenario id: REMS_SF#10				**Purpose**
Parameter: <REM_dispatch>_with_XML_SUB_ACC		Var EV_SET#1 = {XML_SUB_ACC, XML_CONT_CONS, XML_CONS_NOT, XML_CONS_NOT_FAIL }		As scenario REMS_SF#3 but now REMS sends notifications of consignment and one of the REMS consignment notification fails
Parameter: <REMS_notification>_of_Consignment				
Parameter: <REMS_receipt>_with_EV_SET#1				
Sequence of actions				
#	**Sender**	**REMS**	**Receiving side**	
1	Sender sends the original message			
2		Accepts submission. Generates XML_SUB_ACC ERDS evidence		
3		Generates N <REM_dispatch>_with_XML_SUB_ACC		
4		Consigns REM dispatch to the receiving side	N <REM_dispatch>_with_XML_SUB_ACC consigned to receiving side	
5		Generates <REMS_notification>_of_Consignment for N entities	REM dispatch with the original message AND REMS receipt consigned to recipient	
6		Sends N <REMS_notification>_of_Consignment but one fails.		
7		Generates one XML_CONS_NOT_FAIL ERDS evidence for one entity and one XML_CONS_NOT ERDS evidence for N-1 entities	N-1 <REMS_notification>_forConsignment are received successfully; 1 fails	
8		Generates <REMS_receipt>_with_EV_SET#1		
9	Receives <REMS_receipt>_with_EV_SET#1			

Table 3j: Scenarios for intra-REMS operating in Store and Forward style (11/13)

Scenario id: REMS_SF#11				**Purpose**
Parameter: <REM_dispatch>_with_XML_SUB_ACC		Var EV_SET#1 = {XML_SUB_ACC, XML_CONT_CONS, XML_CONS_NOT, 2 XML_CONS_NOT_FAIL }		As scenario REMS_SF#3 but now REMS sends notifications of consignment and two of the REMS consignment notifications fail due to different reasons. This implies the generation of two XM_CONST_NOT_FAIL ERDS evidences
Parameter: <REMS_notification>_of_Consignment				
Parameter: <REMS_receipt>_with_EV_SET#1				
Sequence of actions				
#	**Sender**	**REMS**	**Receiving side**	
1	Sender sends the original message			
2		Accepts submission. Generates XML_SUB_ACC ERDS evidence		
3		Generates N <REM_dispatch>_with_XML_SUB_ACC		
4		Consigns REM dispatch to the receiving side	N <REM_dispatch>_with_XML_SUB_ACC consigned to receiving side	
5		Generates <REMS_notification>_of_Consignment for N entities	REM dispatch with the original message AND REMS receipt consigned to recipient	
6		Sends N <REMS_notification>_of_Consignment but one fails.		
7		Generates 2 XML_CONS_NOT_FAIL ERDS evidences for one entity and one XML_CONS_NOT ERDS evidence for N-1 entities	N-2<REMS_notification>_for Consignment are received successfully; 2 <REMS_notification>_for Consignment fail	
8		Generates <REMS_receipt>_with_EV_SET#1		
9	Receives <REMS_receipt>_with_EV_SET#1			

Table 3k: Scenarios for intra-REMS operating in Store and Forward style (12/13)

Scenario id: REMS_SF#12				**Purpose**
Parameter: <REM_payload>		Var EV_SET#1 = {XML_SUB_ACC, XML_CONT_CONS}		As scenario REMS_SF#3 but now the original message and the ERDS evidence travel in different REM messages to the receiving side
Parameter: <REMS_receipt_1>_with_XML_SUB_ACC				
Parameter: <REMS_receipt_2>_with_XML_EV_SET#1				
Parameter: <REMS_receipt>_with_EV_SET#1				
#	**Sender**	**REMS**	**Receiving side**	
1	Sender sends the original message			
2		Accepts submission. Generates one XML_SUB_ACC ERDS evidence		
3		Generates <REM_payload>		
4		Generates <REMS_receipt_1>_with_XML_SUB_ACC		
5		Consigns <REM_payload> and <REMS_receipt_1>_with_XML_SUB_ACC to the receiving side		
6		Generates one XML_CONT_CONS ERDS evidence	<REM_payload>AND <REMS_receipt_1>_with_XML_SUB_ACC consigned to receiving side	
7		Generates <REMS_receipt_2>_with_EV_SET#1		
8	Receives <REMS_receipt_2>_with_EV_SET#1	Sends it to the sender		

NOTE 3: It is possible to define a set of scenarios covering different types of failures in consignment and/or handover, and/or notifications as for the former test cases where ERDS evidences and original messages travelled together within REM dispatches.

Table 3l: Scenarios for intra-REMS operating in Store and Forward style (13/13)

Scenario id: REMS_SF#13				**Purpose**
Parameter: <REM_payload>				As scenario REMS_SF#3 but now the REMS generates and sends an REMS receipt as soon as it generates the corresponding ERDS Evidence
Parameter: <REMS_receipt>_with_XML_SUB_ACC				
Parameter: <REMS_receipt>_with_XML_CONT_CONS				
Sequence of actions				
#	**Sender**	**REMS**	**Receiving side**	
1	Sender sends original message			
2		Accepts submission. Generates XML_SUB_ACC ERDS evidence		
		Generates <REMS receipt>_with_XML_SUB_ACC and sends it to both sender and receiving side		
		Sends <REMS receipt>_with_XML_SUB_ACC to both sender and receiving side		
	Sender receives <REMS receipt>_with_XML_SUB_ACC		Entities in receiving side receive <REMS receipt>_with_XML_SUB_ACC	
3		Generates <REM_payload>_ for the N receiving entities		
4		Consigns REM payload to receiving side		
5		Generates one XML_CONT_CONS ERDS evidence with details of the N recipients	<REM_payload>_ to receiving side	
6		Generates <REMS_receipt>_with_XML_CONT_CONS		
7		Sends <REMS_receipt>_with_XML_CONT_CONS back to sender		
8	Receives <REMS_receipt>_with_XML_CONT_CONS			

NOTE 4: It is possible to define a set of scenarios where:

1) each time the REMS generates an ERDS evidence, it generates and sends an <ERDS receipt> immediately after; and

2) the original message travels within an <REM payload>.

This set would cover different types of failures in consignment and/or handover, and/or notifications as for the former test cases where ERDS evidences and original messages travelled together within REM dispatches.

5.3.3 Scenarios for Store and Notify style

Table 4 defines a number of scenarios for the case where sender and the entities at receiving side are subscribed to the same REMS operating in Store and Notify style.

Table 4: Scenarios for intra-REMS operating in Store&Notify model of operation (1/4)

Scenario id: REMS_SN#1				**Purpose**
Parameter: <REM_dispatch>_with_XML_SUB_ACC		Var EV_SET#1 = { XML_NOT_F_ACC ERDS, XML_CONS_ACC, XML_CONT_CONS}		First scenario for Store&Notify style, where the REMS asks to receiving side for acceptance, and all the entities at receiving side accept
Parameter: <REMS_notification>_for_Acceptance				
Parameter: <REMS_receipt>_with_EV_SET#1				
Sequence of actions				
#	**Sender**	**REMS**	**Receiving side**	
1	Sender sends original message			
2		Accepts submission. Generates XML_SUB_ACC ERDS evidence		
3		Generates <REMS_notification>_for_Acceptance		
4		Sends <REMS_notification>_for_Acceptance		
5			All entities in receiving side receive one <REMS_notification>_for_Acceptance and answer positively	
6		Generates XML_NOT_F_ACC ERDS evidence		
7		Receives positive responses from receiving side and generates XML_CONS_ACC ERDS evidence for the N entities at receiving side		
8		Generates <REM_dispatch>_with_XML_SUB_ACC		
9		Consigns N <REM_dispatch>_with_XML_SUB_ACC		
10		Generates XML_CONT_CONS ERDS evidence	N <REM_dispatch>_with_XML_SUB_ACC consigned to receiving side	
11		Generates <REMS_receipt>_with_EV_SET#1		
12	Receives <REMS_receipt>_with_EV_SET#1			

Table 4a: Scenarios for intra-REMS operating in Store&Notify model of operation (2/4)

Scenario id: REMS_SN#2				**Purpose**
Parameter: <REM_dispatch>_with_XML_SUB_ACC		Var EV_SET#1 = {XML_SUB_ACC, XML_NOT_F_ACC, XML_CONS_ACC, XML_CONS_REJ, XML_CONT_CONS}		As before but one of the entities at the receiving side does not accept consignment
Parameter: <REMS_notification>_for_Acceptance				
Parameter: <REMS_receipt>_with_EV_SET#1				
Sequence of actions				
#	**Sender**	**REMS**	**Receiving side**	
1	Sender sends original message			
2		Accepts submission. Generates XML_SUB_ACC ERDS evidence		
3		Generates N <REMS_notification>_for_Acceptance		
4		Sends N <REMS_notification>_for_Acceptance		
5		Generates one XML_NOT_F_ACC ERDS evidence for N entities	Each entity receives one <REMS_notification>_for_Acceptance. N-1 answer positively, one answers negatively	
6		Receives N-1 positive answers and one negative answer		
7		Generates one XML_CONS_ACC ERDS evidence for N-1 entities and one XML_CONS_REJ ERDS evidence		
8		Generates one <REM_dispatch>_with_XML_SUB_ACC		
9		Consigns it to the N-1 entities at receiving side		
10		Generates XML_CONT_CONS for N-1 entities	<REM_dispatch>_with_XML_SUB_ACC consigned to N-1 entities at receiving side	
11		Generates <REMS_receipt>_with_EV_SET#1		
12		Sends it to sender		
13	Receives <REMS_receipt>_with_EV_SET#1			

Table 4b: Scenarios for intra-REMS operating in Store&Notify model of operation (3/4)

Scenario id: REMS_SN#3				Purpose
Parameter: <REM_dispatch>_with_XML_SUB_ACC		Var EV_SET#1 = {XML_SUB_ACC, XML_NOT_F_ACC , XML_ACC_REJ_EXP, XML_CONS_ACC, XML_CONT_CONS }		As before but one of the entities at the receiving side does not answer in time
Parameter: <REMS_notification>_for_Acceptance				
Parameter: <REMS_receipt>_with_EV_SET#1				
Sequence of actions				
#	**Sender**	**REMS**	**Receiving side**	
1	Sender sends original message			
2		Accepts submission. Generates XML_SUB_ACC ERDS evidence		
3		Generates N <REMS_notification>_for_Acceptance		
4		Sends <REMS_notification>_for_Acceptance to N entities		
5		Generates one XML_NOT_F_ACC ERDS evidence for N entities	Each entity receives one <REMS_notification>_for_Acceptance. N-1 answer positively, one does not answer in time	
6		Receives N-1 positive answers		
7		When the expiration time is reached generates <REM_dispatch>_with_XML_SUB_ACC for N-1 entities		
		Consigns them to the N-1 entities that have accepted		
		Generates one XML_CONS_ACC ERDS evidence for N-1 entities and one XML_ ACC_REJ_EXP ERDS evidence for one entity	N-1 entities at receiving side receive the <REM_dispatch>_with_XML_SUB_AC C	
8		Generates <REMS_receipt>_with_EV_SET#1		
9		Sends it to the sender		
10	Receives <REMS_receipt>_with_EV_SET#1			

Table 4c: Scenarios for intra-REMS operating in Store&Notify model of operation (4/4)

Scenario id: REMS_SN#4				**Purpose**
Parameter: <REM_dispatch>_with_XML_SUB_ACC		Var EV_SET#1 = { XML_NOT_F_ACC, XML_CONT_HAND }		First scenario for Store&Notify style, where the REMS asks to receiving side for acceptance, and all the entities at receiving side accept
Parameter: <REMS_notification>_for_Acceptance				
Parameter: <REMS_receipt>_with_EV_SET#1				
Sequence of actions				
#	**Sender**	**REMS**	**Receiving side**	
1	Sender sends original message			
2		Accepts submission. Generates XML_SUB_ACC ERDS evidence		
		Generates <REMS Receipt> with XML_SUB_ACC ERDS		
		Sends back <REMS Receipt> with XML_SUB_ACC ERDS evidence to sender		
3	Sender receives <REMS Receipt> with XML_SUB_ACC ERDS evidence	Generates N <REMS_notification>_for Acceptance		
4		SendsN <REMS_notification>_for Acceptance		
5			All entities in receiving side receive one <REMS_notification>_for Acceptance	
6		Generates XML_NOT_F_ACC ERDS evidence for N entities	All entities in receiving side and answer positively to the N <REMS_notification>	
7		Receives positive responses from receiving side	All entities in receiving retrieve the REM dispatch	
8		Generates one XML_CONT_HAND ERDS evidence with details of the N recipients		
9		Generates <REMS_receipt>_with_EV_SET#1		
		Sends <REMS_receipt>_with_EV_SET#1 back to the sender		
10	Receives <REMS_receipt>_with_EV_SET#1			

5.4 Scenarios for 4-corner model

5.4.1 Introduction

The present clause defines test cases for scenarios that take place when the sender and the entities at the receiving side are subscribed to different REMSs but there are not intermediate REMSs between the SREMS and the RREMSs.

Clause 5.4.2 defines test cases when both REMSs operate in Store and Forward style.

Clause 5.4.3 defines test cases when the SREMS operates Store and Forward style and the RREMS operates Store and Notify style.

Clause 5.4.4 defines test cases when the SREMS operates Store and Notify style and the RREMS operates Store and Forward style.

5.4.2 Scenarios for Store&Forward to Store&Forward

Table 5 defines a number of scenarios for the case where SREMS and RREMS operate both Store and Forward style and are NOT the same REMS.

The scenarios are based on scenarios in Table 3 adding the relay acceptance and relay rejection events at the some of the RREMSs and their corresponding ERDS evidences.

For the sake of simplicity, it will suppose that all the entities at receiving side are served by the same RREMS. It could be possible to use the templates defined in the present document for defining scenarios where the aforementioned entities are served by different RREMSs.

Table 5: Scenarios for SREMS and RREMSs operating Store&Forward (1/5)

Scenario id: SREMS_SF_RREMS_SF#1					Purpose
Parameter: <REMS_dispatch>_with_XML_SUB_ACC			Var EV_SET#1 = {XML_SUB_ACC, XML_REL_REJ}		The simplest scenario: the RREMS rejects the REM dispatch relayed by the SREMS and sends back a REM receipt with one RelayRejection ERDS evidence
Parameter: <REMS_receipt_1>_with_XML_REL_REJ					
Parameter: <REMS_receipt_2>_with_EV_SET#1					
Sequence of actions					
#	**Sender**	**SREMS**	**RREMS**	**Receiving side**	
1	Sender sends original message				
2		Accepts submission. Generates XML_SUB_ACC ERDS evidence			
3		Generates <REM_dispatch>_with_XML_SUB_ACC			
4		Relays the <REM_dispatch>_with_XML_SUB_ACC to the RREMS			
5			The RREMSs Rejects the <REM_dispatch>_with_XML_SUB_ACC		
6			Generates XML_REL_REJ ERDS evidence		
7			Generates <REMS_receipt_1>_with_XML_REL_REJ		
8			Sends <REMS_receipt_1>_with_XML_REL_REJ to SREMS		
9		Receives <REMS_receipt_1>_with_XML_REL_REJ			
10		Generates <REMS_receipt_2>_with_EV_SET#1			
11		Sends it back to sender			
12	Sender receives <REMS_receipt_2>_with_EV_SET#1				

Table 5a: Scenarios for SREMS and RREMSs operating Store&Forward (2/5)

Scenario id: SREMS_SF_RREMS_SF#2					**Purpose**
Parameter: <REM_dispatch_1>_with_XML_SUB_ACC					The simplest successful scenario: the SREMS accepts the submission of the original message, generates one REM dispatch and relays to RREMS. This accepts relay, builds its own REM dispatch and delivers it to the N recipients in receiving side. SREMS generates and sends back to the sender a REM receipt with one SubmissionAcceptance ERDS evidence, one RelayAcceptance, and one ContentConsignment ERDS evidence Clause 4.3.2.1 of ETSI EN 319 532-1 [3] shows a variation of this scenario where RREMS sends XML_REL_ACC and XML_CONT_CONS in different REM receipts
Parameter <REMS_receipt_1>_with_XML_SUB_ACC					
Parameter: <REMS_receipt_2>_with_XML_REL_ACC					
Parameter: <REM_dispatch_2>_with_XML_SUB_ACC					
Parameter: <REMS_receipt_3>_with_XML_CONT_CONS					
Sequence of actions					
#	**Sender**	**SREMS**	**RREMS**	**Receiving side**	
1	Sender sends original message				
2		Accepts submission. Generates XML_SUB_ACC ERDS evidence			
3		Generates <REMS_receipt_1>_with_XML_SUB_ACC and sends it to the sender			
4	Receives <REMS_receipt_1>_with_XML_SUB_ACC	Generates <REM_dispatch_1>_with_XML_SUB_ACC			
5		Relays it to RREMS REM dispatch to recipient			
6			Accepts it and generates XML_REL_ACC ERDS evidence		
7			Generates <REMS_receipt_2>_with_XML_REL_ACC		
8			Sends <REMS_receipt_2>_with_XML_REL_ACC to SREMS		
9		Receives <REMS_receipt_2>_with_XML_REL_ACC	Generates <REM_dispatch_3>_with_XML_SUB_ACC		
10			Consigns it to the receiving side		
11			Generates XML_CONT_CONS ERDS evidence	<REM_dispatch_2>_with_XML_SUB_ACC consigned to the N entities in receiving side	
12			Generates <REMS_receipt_3>_with_XML_CONT_CONS		
13			Sends it back to the SREMS		
14		Receives <REMS_receipt_3>_with_XML_CONT_CONS and sends it back to the sender			

Scenario id: SREMS_SF_RREMS_SF#2					Purpose
15	Receives <REMS_receipt_3>_with_XML_CONT_CONS				

Table 5b: Scenarios for SREMS and RREMSs operating Store&Forward (3/5)

Scenario id: SREMS_SF_RREMS_SF#3					Purpose
Parameter: <REM_dispatch_1>_with_XML_SUB_ACC			Var EV_SET#1 = {REL_ACC, CONT_CONS}		As scenario SREMS_SF_RREMS_SF#2 but now finalized with hand over and RelayAcceptance and ContentConsignment travel together in the same REM receipt back to SREMS
Parameter: <REM_dispatch_2>_with_XML_SUB_ACC					
Parameter: <REMS_receipt_1>_with_XML_SUB_ACC					
Parameter: <REMS_receipt_2>_with_EV_SET#1					
Parameter: <REMS_receipt_3>_with_XML_CONT_HAND					
Sequence of actions					
#	Sender	REMS	RREMS	Receiving side	
1	Sender sends original message				
2		Accepts submission. Generates XML_SUB_ACC ERDS evidence			
3		Generates <REMS_receipt_1>_with_XML_SUB_ACC and sends it to the sender			
4	Receives <REMS_receipt_1>_with_XML_SUB_ACC	Generates <REM_dispatch_1>_with_XML_SUB_ACC			
5		Relays <REM_dispatch_1>_with_XML_SUB_ACC TO RREMS			
6			Accepts it and generates XML_REL_ACC ERDS evidence		
7			Generates <REM_dispatch_2>_with_XML_SUB_ACC		
8			Consigns it to the receiving side		
9			Generates XML_CONT_CONS ERDS evidence	<REM_dispatch_2>_with_XML_SUB_ACC is consigned to all the entities at receiving side	
10			Generates <REMS_receipt_2>_with_EV_SET#1		
11			Send <REMS_receipt_2>_with_EV_SET#1 to SREMS	Entities in receiving side retrieve user content	

Scenario id: SREMS_SF_RREMS_SF#3					Purpose
12		Receives <REMS_receipt_2>_with_EV_SET#1 and sends it back to the sender	Generates XML_CONT_HAND ERDS evidence		
13	Receives <REMS_receipt_2>_with_EV_SET#1		Generates <REMS_receipt_3>_with_XML_CONT_HAND		
14			Sends <REMS_receipt_3>_with_XML_CONT_HAND to SREMS		
15		Receives <REMS_receipt_3>_with_XML_CONT_HAND and sends it back to the sender			
16	Receives <REMS_receipt_3>_with_XML_CONT_HAND				

Table 5c: Scenarios for SREMS and RREMSs operating Store&Forward (4/5)

Scenario id: SREMS_SF_RREMS_SF#4					Purpose
Parameter: <REM_dispatch_1>_with_XML_SUB_ACC			Var EV_SET#1 = {XNL_REL_ACC,XML_CON_CONS}		As the previous scenario but now one of the handing over fails. Hereinafter, the scenarios do not show handing over, but only consignment. However, a set of scenarios including handing over could be easily built based on them
Parameter: <REM_dispatch_2>_with_XML_SUB_ACC			Var EV_SET#2 = {XML_CONT_HAND_FAIL}		
Parameter: <REMS_receipt_1>_with_XML_SUB_ACC					
Parameter: <REMS_receipt_2>_with_EV_SET#1					
Parameter: <REMS_receipt_3>_with_EV_SET#2					
Sequence of actions					
#	**Sender**	**SREMS**	**RREMS**	**Receiving side**	
1	Sender sends original message				
2		Accepts submission. Generates XML_SUB_ACC ERDS evidence			
3		Generates <REMS_receipt_1>_with_XML_SUB_ACC and sends it to the sender			
4	Receives <REMS_receipt_1>_with_XML_SUB_ACC	Generates <REM_dispatch_1>_with_XML_SUB_ACC			
5		Relays <REM_dispatch_1>_with_XML_SUB_ACC to RREMS			
6			Accepts it and generates XML_REL_ACC ERDS evidence		
7			Generates <REM_dispatch_2>_with_XML_SUB_ACC		

Scenario id: SREMS_SF_RREMS_SF#4					Purpose
8			Consigns it to the receiving side		
9			Generates XML_CONT_CONS ERDS evidence	<REM_dispatch_2>_with_XML_SUB_ACC is consigned to all the entities at receiving side	
10			Generates <REMS_receipt_2>_with_EV_SET#1		
11			Send <REMS_receipt_2>_with_EV_SET#1 to SREMS	Entities in receiving side retrieve user content. N-1 succeed. One fails.	
12		Receives <REMS_receipt_2>_with_EV_SET#1 and sends it back to the sender	Generates one XML_CONT_HAND ERDS evidence for N-1 entities and one XML_CONT_HAND_FAIL ERDS evidence for one entity		
13	Receives <REMS_receipt_2>_with_EV_SET#1		Generates <REMS_receipt_3>_with_EV_SET#2		
14			Sends <REMS_receipt_3>_with_EV_SET#2 to SREMS		
15		Receives <REMS_receipt_3>_with_EV_SET#2 and sends it back to the sender			
16	Receives <REMS_receipt_3>_with_EV_SET#2				

Table 5d: Scenarios for SREMS and RREMSs operating Store&Forward (5/5)

Scenario id: SREMS_SF_RREMS_SF#5					Purpose
Parameter: <REM_dispatch_1>_with_XML_SUB_ACC			Var EV_SET#1 = {CONT_CONS, CON_CONS_FAIL}		As scenario SREMS_SF_RREMS_SF#2 but now one of the REM dispatch consignments fails
Parameter: <REM_dispatch_2>_with_XML_SUB_ACC					
Parameter: <REMS_receipt_1>_with_XML_REL_ACC					
Parameter: <REMS_receipt_2>_with_EV_SET#1					
Sequence of actions					
#	Sender	SREMS	RREMS	Receiving side	
1	Sender sends original message				
2		Accepts submission. Generates XML_SUB_ACC ERDS evidence			
3		Generates <REM_dispatch_1>_with_XML_SUB_ACC			
4		Relays it to RREMS			

Scenario id: SREMS_SF_RREMS_SF#5					Purpose
5			Accepts it and generates XML_REL_ACC ERDS evidence		
6			Generates <REMS_receipt_1>_with_XML_REL_ACC		
7			Sends <REMS_receipt_1>_with_XML_REL_ACC to SREMS		
8		Receives <REMS_receipt_1>_with_XML_REL_ACC	Generates <REM_dispatch_2>_with_XML_SUB_ACC		
9			Consigns it to receiving side. One fails.		
10				<REM_dispatch_2>_with_XML_SUB_ACC consigned to N-1 entities in receiving side. The other consignment fails	
11			Generates one XML_CONT_CONS ERDS evidence related to N-1 entities Generates one XML_CONT_CONS_FAIL related to one entity		
12			Generates <REMS_receipt_2>_with_EV_SET#1		
13			Sends it back to the SREMS		
14		Receives <REMS_receipt_2>_with_EV_SET#1 and sends it back to the sender			
15	Receives <REMS_receipt_2>_with_EV_SET#1				

5.4.3 Scenarios for Store&Forward to Store&Notify

Table 6 defines a number of scenarios for the case where SREMS operates Store and Forward and RREMS operates Store and Notify.

The scenarios are based on scenarios in Table 4 adding the relay acceptance and relay rejection events at the some of the RREMSs and their corresponding ERDS evidences.

For the sake of simplicity, it will suppose that all the entities at receiving side are served by the same RREMS. It could be possible to use the templates defined in the present document for defining scenarios where the aforementioned entities are served by different RREMSs.

Table 6: Scenarios where SREMS operates Store&Forward and RREMSs operate Store&Notify (1/5)

Scenario id: SREMS_SF_RREMS_SN#1					Purpose
Parameter: <REM_dispatch_1>_with_XML_SUB_ACC			Var EV_SET#1 = {NOT_F_ACC, CONS_ACC, CONT_CONS, CONT_HAND}		First scenario for Store&Notify style, where the REMS asks to all the entities at receiving side for acceptance, and all the entities at receiving side accept. Similar scenarios to the ones present in this table could have been defined where each REMS receipt contains only one ERDS evidence, instead several. Scenario SREMS_SF_IREMS_SF_RREMS_SF#1 in Table 8 of clause 5.5.2 is an example
Parameter: <REM_dispatch_2>_with_XML_SUB_ACC					
Parameter: <REMS_receipt_1>_with_XML_REL_ACC					
Parameter: <REMS_notification>_for_Acceptance					
Parameter: <REMS_receipt_2>_with_EV_SET#1					
Sequence of actions					
#	**Sender**	**SREMS**	**RREMS**	**Receiving side**	
1	Sender sends original message				
2		Accepts submission. Generates XML_SUB_ACC ERDS evidence			
3		Generates <REM_dispatch_1>_with_XML_SUB_ACC			
4		Relays <REM_dispatch_1>_with_XML_SUB_ACC to RREMS			
5			Accepts <REM_dispatch_1>_with_XML_SUB_ACC		
6			Generates XML_REL_ACC ERDS evidence		
7			Generates <REMS_receipt_1>_with_XML_REL_ACC		
8			Sends <REMS_receipt_1>_with_XML_REL_ACC back to SREMS		
9		Receives <REMS_receipt_1>_with_XML_REL_ACC	Generates <REMS_notification>_for_Acceptance		
10			Sends <REMS_notification>_for_Acceptance to receiving side		
11			Generates XML_NOT_F_ACC ERDS evidence	All entities in receiving side answer positively	
12			Generates XML_CONS_ACC ERDS evidence		
13			Generates <REM_dispatch_2>_with_XML_SUB_ACC		

Scenario id: SREMS_SF_RREMS_SN#1					Purpose
14			Sends <REM_dispatch_2>_with_XML_SUB_ACC to receiving side		
15				<REM_dispatch_2>_ with_XML_SUB_ACC consigned to all entities in receiving side	
16			Generates XML_CONT_CONS ERDS evidence	All the entities retrieve user content	
17			Generates XML_CONT_HAND ERDS evidence		
18			Generates <REMS_receipt_2>_with_EV_SET#1		
19			Sends <REMS_receipt_2>_with_EV_SET#1 to SREMS		
20		Receives <REMS_receipt_2>_with_EV_SET#1			
21		Sends <REMS_receipt_2>_with_XML_XML_EV_SET#1 back to sender			
22	Receives <REMS_receipt_2>_with_ EV_SET#1				

Table 6a: Scenarios where SREMS operates Store&Forward and RREMSs operate Store&Notify (2/5)

Scenario id: SREMS_SF_RREMS_SN#2					Purpose
Parameter: <REM_dispatch_1>_with_XML_SUB_ACC			Var EV_SET#1 = { NOT_F_ACC, CONS_ACC, CONS_REJ, CONT_CONS, CONT_HAND }		As previous scenario but one of the entities at the receiving side does not accept consignment
Parameter: <REM_dispatch_2>_with_XML_SUB_ACC					
Parameter: <REMS_receipt_1>_with_XML_REL_ACC					
Parameter: <REMS_notification>_for_Acceptance					
Parameter: <REMS_receipt_2>_with_EV_SET#1					
Sequence of actions					
#	Sender	SREMS	RREMS	Receiving side	
1	Sender sends original message				
2		Accepts submission. Generates XML_SUB_ACC ERDS evidence			
3		Generates <REM_dispatch_1>_with_XML_SUB_ACC			

Scenario id: SREMS_SF_RREMS_SN#2					**Purpose**
4		Relays <REM_dispatch_1>_with_XML_SUB_ACC to RREMS			
5			Accepts <REM_dispatch_1>_with_XML_SUB_ACC		
6			Generates XML_REL_ACC ERDS evidence		
7			Generates <REMS_receipt_1>_with_XML_REL_ACC		
8			Sends <REMS_receipt_1>_with_XML_REL_ACC back to SREMS		
9		Receives <REMS_receipt_1>_with_XML_REL_ACC	Generates <REMS_notification>_for_Acceptance		
10			Sends <REMS_notification>_for_Acceptance to receiving side		
11			Generates XML_NOT_F_ACC ERDS evidence for all N entities	N-1 entities in receiving side answer positively. One answers negatively	
12			Generates XML_CONS_ACC ERDS evidence for N-1 entities and one XML_CONS_REJ ERDS evidence for one entity		
13			Generates <REM_dispatch_2>_with_XML_SUB_ACC		
14			Sends <REM_dispatch_2>_with_XML_SUB_ACC to N-1 accepting entities at receiving side		
15				<REM_dispatch_2>_ with_XML_SUB_ACC consigned to N-1 entities in receiving side	
16			Generates one XML_CONT_CONS ERDS evidence for N-1 entities	N-1 entities retrieve user content	
17			Generates one XML_CONT_HAND ERDS evidence for N-1 entities		
18			Generates <REMS_receipt_2>_with_EV_SET#1		

Scenario id: SREMS_SF_RREMS_SN#2					Purpose
19			Sends <REMS_receipt_2>_with_EV_SET#1 to SREMS		
20		Receives <REMS_receipt_2>_with_EV_SET#1			
21		Sends <REMS_receipt_2>_with_XML_XML_EV_SET#1 back to sender			
22	Receives <REMS_receipt_2>_with_EV_SET#1				

Table 6b: Scenarios where SREMS operates Store&Forward and RREMSs operate Store&Notify (3/5)

Scenario id: SREMS_SF_RREMS_SN#3					Purpose
Parameter: <REM_dispatch_1>_with_XML_SUB_ACC			Var EV_SET#1 = { NOT_F_ACC, CONS_ACC, ACC_REJ_EXP, CONT_CONS, CONT_HAND }		As previous scenario but one of the entities at the receiving side does not answer in time
Parameter: <REM_dispatch_2>_with_XML_SUB_ACC					
Parameter: <REMS_receipt_1>_with_XML_REL_ACC					
Parameter: <REMS_notification>_for_Acceptance					
Parameter: <REMS_receipt_2>_with_EV_SET#1					
Sequence of actions					
#	**Sender**	**SREMS**	**RREMS**	**Receiving side**	
1	Sender sends original message				
2		Accepts submission. Generates XML_SUB_ACC ERDS evidence			
3		Generates <REM_dispatch_1>_with_XML_SUB_ACC			
4		Relays <REM_dispatch_1>_with_XML_SUB_ACC to RREMS			
5			Accepts <REM_dispatch_1>_with_XML_SUB_ACC		
6			Generates XML_REL_ACC ERDS evidence		
7			Generates <REMS_receipt_1>_with_XML_REL_ACC		
8			Sends <REMS_receipt_1>_with_XML_REL_ACC back to SREMS		

Scenario id: SREMS_SF_RREMS_SN#3					Purpose
9		Receives <REMS_receipt_1>_with_XML_REL_ACC	Generates <REMS_notification>_for_Acceptance		
10			Sends <REMS_notification>_for_Acceptance to receiving side		
11			Generates one XML_NOT_F_ACC ERDS evidence for all N entities	N-1 entities in receiving side answer positively. One does not answer in time	
12			Generates one XML_CONS_ACC ERDS evidence for N-1 entities and one XML_ACC_REJ_EXP ERDS evidence for one entity		
13			Generates <REM_dispatch_2>_with_XML_SUB_ACC		
14			Sends <REM_dispatch_2>_with_XML_SUB_ACC to N-1 accepting entities at receiving side		
23				<REM_dispatch_2>_ with_XML_SUB_ACC consigned to N-1 entities in receiving side	
15			Generates one XML_CONT_CONS ERDS evidence for N-1 entities	N-1 entities retrieve user content	
16			Generates one XML_CONT_HAND ERDS evidence for N-1 entities		
17			Generates <REMS_receipt_2>_with_EV_SET#1		
18			Sends <REMS_receipt_2>_with_EV_SET#1 to SREMS		
19		Receives <REMS_receipt_2>_with_EV_SET#1			
20		Sends <REMS_receipt_2>_with_XML_XML_EV_SET#1 back to sender			
21	Receives <REMS_receipt_2>_with_ EV_SET#1				

Table 6c: Scenarios where SREMS operates Store&Forward and RREMSs operate Store&Notify (4/5)

Scenario id: SREMS_SF_RREMS_SN#4					Purpose
Parameter: <REM_dispatch_1>_with_XML_SUB_ACC			Var EV_SET#1 = { NOT_F_ACC, CONS_ACC, CONT_CONS, CONT_CONS_FAIL, CONT_HAND }		As first scenario in the present table, but one of the consignments fails
Parameter: <REM_dispatch_2>_with_XML_SUB_ACC					
Parameter: <REMS_receipt_1>_with_XML_REL_ACC					
Parameter: <REMS_notification>_for_Acceptance					
Parameter: <REMS_receipt_2>_with_EV_SET#1					
Sequence of actions					
#	**Sender**	**SREMS**	**RREMS**	**Receiving side**	
1	Sender sends original message				
2		Accepts submission. Generates XML_SUB_ACC ERDS evidence			
3		Generates <REM_dispatch_1>_with_XML_SUB_ACC			
4		Relays <REM_dispatch_1>_with_XML_SUB_ACC to RREMS			
5			Accepts <REM_dispatch_1>_with_XML_SUB_ACC		
6			Generates XML_REL_ACC ERDS evidence		
7			Generates <REMS_receipt_1>_with_XML_REL_ACC		
8			Sends <REMS_receipt_1>_with_XML_REL_ACC back to SREMS		
9		Receives <REMS_receipt_1>_with_XML_REL_ACC	Generates <REMS_notification>_for_Acceptance		
10			Sends <REMS_notification>_for_Acceptance to receiving side		
11			Generates one XML_NOT_F_ACC ERDS evidence for all N entities	All the entities in receiving side answer positively.	
12			Generates one XML_CONS_ACC ERDS evidence for all the N entities in receiving side		
13			Generates <REM_dispatch_2>_with_XML_SUB_ACC		

Scenario id: SREMS_SF_RREMS_SN#4					Purpose
14			Successfully consigns <REM_dispatch_2>_with_XML_SUB_ACC to N-1 entities at receiving side, one consignment fails		
15				<REM_dispatch_2>_ with_XML_SUB_ACC consigned to N-1 entities in receiving side. One consignment fails	
16			Generates one XML_CONT_CONS ERDS evidence for N-1 entities and one XML_CONT_CONS_FAIL for one entity	N-1 entities retrieve user content	
17			Generates one XML_CONT_HAND ERDS evidence for N-1 entities		
18			Generates <REMS_receipt_2>_with_EV_SET#1		
19			Sends <REMS_receipt_2>_with_EV_SET#1 to SREMS		
20		Receives <REMS_receipt_2>_with_EV_SET#1			
21		Sends <REMS_receipt_2>_with_XML_XML_EV_SET#1 back to sender			
22	Receives <REMS_receipt_2>_with_ EV_SET#1				

Table 6d: Scenarios where SREMS operates Store&Forward and RREMSs operate Store&Notify (5/5)

Scenario id: SREMS_SF_RREMS_SN#5					Purpose
Parameter: <REM_dispatch_1>_with_XML_SUB_ACC		Var EV_SET#1 = {NOT_F_ACC, CONS_ACC, CONT_CONS, CONT_HAND, CONT_HAND_FAIL}			As first scenario in the present table, but one of the handovers fails
Parameter: <REM_dispatch_2>_with_XML_SUB_ACC					
Parameter: <REMS_receipt_1>_with_XML_REL_ACC					
Parameter: <REMS_notification>_for_Acceptance					
Parameter: <REMS_receipt_2>_with_EV_SET#1					
Sequence of actions					
#	**Sender**	**SREMS**	**RREMS**	**Receiving side**	
1	Sender sends original message				
2		Accepts submission. Generates XML_SUB_ACC ERDS evidence			

Scenario id: SREMS_SF_RREMS_SN#5					Purpose
3		Generates <REM_dispatch_1>_with_XML_SUB_ACC			
4		Relays <REM_dispatch_1>_with_XML_SUB_ACC to RREMS			
5			Accepts <REM_dispatch_1>_with_XML_SUB_ACC		
6			Generates XML_REL_ACC ERDS evidence		
7			Generates <REMS_receipt_1>_with_XML_REL_ACC		
8			Sends <REM_receipt_1>_with_XML_XML_REL_ACC back to SREMS		
9		Receives <REMS_receipt_1>_with_XML_REL_ACC	Generates <REMS_notification>_for_Acceptance		
10			Sends <REMS_notification>_for_Acceptance to receiving side		
11			Generates one XML_NOT_F_ACC ERDS evidence for all the N entities	All the entities in receiving side answer positively.	
12			Generates one XML_CONS_ACC ERDS evidence for all the N entities in receiving side		
13			Generates <REM_dispatch_2>_with_XML_SUB_ACC		
14			Sends <REM_dispatch_2>_with_XML_SUB_ACC to all entities at receiving side		
15				<REM_dispatch_2>_with_XML_SUB_ACC consigned to receiving side	
16			Generates one XML_CONT_CONS ERDS evidence for all the N entities	N-1 entities retrieve user content. One fails	
17			Generates one XML_CONT_HAND ERDS evidence for N-1 entities and one XML_CONT_HAND_FAIL for one entity		
18			Generates <REMS_receipt_2>_with_EV_SET#1		

Scenario id: SREMS_SF_RREMS_SN#5					Purpose
19			Sends <REMS_receipt_2>_with_EV_SET#1 to SREMS		
20		Receives <REMS_receipt_2>_with_EV_SET#1			
21		Sends <REMS_receipt_2>_with_XML_XML_EV_SET#1 back to sender			
22	Receives <REMS_receipt_2>_with_EV_SET#1				

5.4.4 Scenarios for Store&Notify to Store&Forward

Table 7 defines a number of scenarios for the case where SREMS operates Store and Notify and RREMS operates Store and Forward.

For the sake of simplicity, it will suppose that all the entities at receiving side are served by the same RREMS. It could be possible to use the templates defined in the present document for defining scenarios where the aforementioned entities are served by different RREMSs.

Table 7: Scenarios where SREMS operates Store&Notify and RREMSs operate Store&Forward (1/5)

Scenario id: SREMS_SN_RREMS_SF#1					**Purpose**
Parameter: <REM_dispatch_1>_with_XML_SUB_ACC		Var EV_SET#1 = {XML_CONT_CONS, XML_CONT_HAND }			Successful scenario where the REM dispatch is successfully consigned to all the entities in receiving side and all the entities successfully retrieve it. This scenario is slightly different than the one shown in clause 4.3.2.3 of ETSI EN 319 532-1 [3] because it groups the two last ERDS evidence generated by RREMS in one unique REMS receipt
Parameter: <REMS_notification_1>_for_Acceptance					
Parameter: <REMS_receipt_1>_with_XML_REL_ACC					
Parameter: <REMS_notification_2>_for_Acceptance					
Parameter: <REMS_receipt_2>_with_XML_NOT_F_ACC					
Parameter: <REMS_receipt_3>_with_XML_CONS_ACC					
Parameter: <REMS_receipt_4>_with_XML_REL_ACC					
Parameter: <REMS_receipt_5>_with_EV_SET#1					
Sequence of actions					
#	**Sender**	**SREMS**	**RREMS**	**Receiving side**	
1	Sender sends original message				
2		Accepts submission. Generates XML_SUB_ACC ERDS evidence			
3		Generates <REM_dispatch_1>_with_XML_SUB_ACC and stores it internally Generates <REMS_notification_1>_for_Acceptance			
4		Relays <REMS_notification_1>_for_Acceptance			
5			Accepts <REMS_notification_1>_for_Acceptance		
6			Generates XML_REL_ACC ERDS evidence		
7			Generates <REMS_receipt_1>_with_XML_REL_ACC		
			Sends <REMS_receipt_1>_with_XML_XML_REL_ACC back to SREMS		
8		Receives <REMS_receipt_1>_with_XML_REL_ACC	Generates <REMS_notification_2>_for_Acceptance		
9			Sends <REMS_notification_2>_for_Acceptance to receiving side		
10			Generates XML_NOT_F_ACC ERDS evidence		
11			Generates <REMS_receipt_2>_with_XML_NOT_F_ACC		

Scenario id: SREMS_SN_RREMS_SF#1					Purpose
12			Sends <REMS_receipt_2>_with_XML_NOT_F_ACC back to SREMS		
13		Receives <REMS_receipt_2>_with_XML_NOT_F_ACC			
14				All the entities at receiving side answer positively to SREMS	
15		Generates XML_CONS_ACC ERDS evidence for all entities at receiving side			
16		Relays <REM_dispatch_1>_with_XML_SUB_ACC to RREMS			
17		Generates <REMS_receipt_3>_with_XML_CONS_ACC	Receives <REM_dispatch_1>_with_XML_SUB_ACC		
18		Sends it to sender	Generates XML_REL_ACC ERDS evidence		
19	Receives <REMS_receipt_3>_with_XML_CONS_ACC		Generates <REMS_receipt_3>_with_XML_REL_ACC		
20			Sends it back to SREMS		
21		Receives <REMS_receipt_4>_with_XML_REL_ACC	Generates <REM_dispatch_2>_with_XML_SUB_ACC		
22			Consigns it to receiving side		
23			Generates XML_CONT_CONS ERDS evidence for all entities at receiving side	All the entities at receiving side retrieve the user content	
24			Generates XML_CONT_HAND ERDS evidence for all entities at receiving side		
25			Generates <REMS_receipt_5>_with_EV_SET#1		
26			Sends <REMS_receipt_5>_with_EV_SET#1 to SREMS		
27		Receives <REMS_receipt_5>_with_EV_SET#1			
28		Sends <REMS_receipt_5>_with_EV_SET#1 back to sender			

Scenario id: SREMS_SN_RREMS_SF#1					Purpose
29	Receives <REMS_receipt_5>_with_EV_SET#1				

Table 7a: Scenarios where SREMS operates Store&Notify and RREMSs operate Store&Forward (2/5)

Scenario id: SREMS_SN_RREMS_SF#2					Purpose
Parameter: <REM_dispatch_1>_with_XML_SUB_ACC			Var EV_SET#1 = { CONS_ACC, CONS_REJ }		As previous scenario but now one of the entities at receiving side rejects the consignment
Parameter: <REMS_notification_1>_for_Acceptance			Var EV_SET#2 = {CONT_CONS, CONT_HAND}		
Parameter: <REMS_notification_2>_for_Acceptance					
Parameter: <REMS_receipt_1>_with_XML_NOT_F_ACC					
Parameter: <REMS_receipt_2>_with_XML_EV_SET#1					
Parameter: <REMS_receipt_3>_with_XML_XML_REL_ACC					
Parameter: <REMS_receipt_4>_with_EV_SET#2					
Sequence of actions					
#	**Sender**	**SREMS**	**RREMS**	**Receiving side**	
1	Sender sends original message				
2		Accepts submission. Generates XML_SUB_ACC ERDS evidence			
3		Generates <REM_dispatch_1>_with_XML_SUB_ACC and stores it internally			
4		Relays <REMS_notification_1>_for_Acceptance			
5		Receives <REMS_receipt_1>_with_XML_REL_ACC	Generates <REMS_notification_2>_for_Acceptance		
6			Sends <REMS_notification_2>_for_Acceptance to receiving side		
7			Generates XML_NOT_F_ACC ERDS evidence		
8			Generates <REMS_receipt_1>_with_XML_NOT_F_ACC		
9			Sends <REMS_receipt_1>_with_XML_NOT_F_ACC back to SREMS		
10		Receives <REMS_receipt_1>_with_XML_NOT_F_ACC			

Scenario id: SREMS_SN_RREMS_SF#2					Purpose
11				N-1 entities at receiving side answer positively to SREMS. One answers negatively to SREMS.	
12		Generates one XML_CONS_ACC ERDS evidence for N-1 accepting entities, generates one XML_CONS_REJ evidence for the one rejecting entity.			
13		Relays <REM_dispatch_1>_with_XML_SUB_ACC to RREMS for N-1 accepting entities			
14		Generates <REMS_receipt_2>_with_XML_EV_SET#1	Receives <REM_dispatch_1>_with_XML_SUB_ACC		
15		Sends <REMS_receipt_2>_with_XML_EV_SET#1 back to sender	Generates XML_REL_ACC ERDS evidence		
16	Receives <REMS_receipt_2>_with_XML_EV_SET#1		Generates <REMS_receipt_3>_with_XML_REL_ACC		
17			Sends it back to SREMS		
18		Receives <REMS_receipt_3>_with_XML_XML_REL_ACC	Generates <REM_dispatch_2>_with_XML_SUB_ACC		
19			Consigns it to receiving side for N-1 accepting entities		
20			Generates one XML_CONT_CONS ERDS evidence for N-1 accepting entities	N-1 entities at receiving side retrieve the user content	
21			Generates one XML_CONT_HAND ERDS evidence for N-1 accepting entities		
22			Generates <REMS_receipt_4>_with_EV_SET#2		
23			Sends <REMS_receipt_4>_with_EV_SET#2 to SREMS		
24		Receives <REMS_receipt_4>_with_EV_SET#2			
25		Sends <REMS_receipt_4>_with_EV_SET#2 back to sender			
26	Receives <REMS_receipt_4>_with_EV_SET#2				

Table 7b: Scenarios where SREMS operates Store&Notify and RREMSs operate Store&Forward (3/5)

Scenario id: SREMS_SN_RREMS_SF#3					**Purpose**
Parameter: <REM_dispatch_1>_with_XML_SUB_ACC			EV_SET#1 = {CONT_CONS, CONT_HAND}		As the first scenario in the present table but now one of the entities at receiving side does not answer in time to the notification for acceptance of SREMS
Parameter: <REMS_notification_1>_for_Acceptance					
Parameter: <REMS_notification_2>_for_Acceptance					
Parameter: <REMS_receipt_1>_with_XML_NOT_F_ACC					
Parameter: <REMS_receipt_2>_with_XML_CONS_ACC					
Parameter: <REMS_receipt_3>_with_XML_XML_REL_ACC					
Parameter: <REMS_receipt_4>_with_EV_SET#1					
Sequence of actions					
#	**Sender**	**SREMS**	**RREMS**	**Receiving side**	
1	Sender sends original message				
2		Accepts submission. Generates XML_SUB_ACC ERDS evidence			
3		Generates <REM_dispatch_1>_with_XML_SUB_ACC and stores it internally			
4		Relays <REMS_notification_1>_for_Acceptance			
5			Accepts <REMS_notification_1>_for_Acceptance		
6		Receives <REMS_receipt_1>_with_XML_REL_ACC	Generates <REMS_notification_2>_for_Acceptance		
7			Sends <REMS_notification_2>_for_Acceptance to receiving side		
8			Generates XML_NOT_F_ACC ERDS evidence		
9			Generates <REMS_receipt_1>_with_XML_NOT_F_ACC		
10			Sends <REMS_receipt_1>_with_XML_NOT_F_ACC back to SREMS		
11		Receives <REMS_receipt_1>_with_XML_NOT_F_ACC			
12				N-1 entities at receiving side answer positively to SREMS. One does not answer in time to SREMS	

Scenario id: SREMS_SN_RREMS_SF#3					Purpose
13		Generates XML_CONS_ACC ERDS evidence for N-1 accepting entities, generates XML_ACC_REJ_EXP ERDS evidence for the one entity which did not respond in time.			
14		Relays <REM_dispatch_1>_with_XML_SUB_ACC to RREMS for N-1 accepting entities			
15		Generates <REMS_receipt_2>_with_XML_CONS_ACC	Receives <REM_dispatch_1>_with_XML_SUB_ACC		
16		Sends <REMS_receipt_2>_with_XML_CONS_ACC back to sender	Generates XML_REL_ACC ERDS evidence		
17	Receives <REMS_receipt_2>_with_XML_CONS_ACC		Generates <REMS_receipt_3>_with_XML_REL_ACC		
18			Sends it back to SREMS		
19		Receives <REMS_receipt_3>_with_XML_XML_REL_ACC	Generates <REM_dispatch_2>_with_XML_SUB_ACC		
20			Consigns it to receiving side for N-1 accepting entities		
21			Generates XML_CONT_CONS ERDS evidence for N-1 accepting entities	N-1 entities at receiving side retrieve the user content	
22			Generates XML_CONT_HAND ERDS evidence for N-1 accepting entities		
23			Generates <REMS_receipt_4>_with_EV_SET#1		
24			Sends <REMS_receipt_4>_with_EV_SET#1 to SREMS		
25		Receives <REMS_receipt_4>_with_EV_SET#1			
26		Sends <REMS_receipt_4>_with_EV_SET#1 back to sender			
27	Receives <REMS_receipt_>_with_EV_SET#21				

Table 7c: Scenarios where SREMS operates Store&Notify and RREMSs operate Store&Forward (4/5)

Scenario id: SREMS_SN_RREMS_SF#4					**Purpose**
Parameter: <REM_dispatch_1>_with_XML_SUB_ACC			Var EV_SET#1 = {CONT_CONS, CONT_CONS_FAIL }		As the first scenario in the present table but now one of the consignments to the receiving side fails
Parameter: <REMS_notification_1>_for_Acceptance					
Parameter: <REMS_receipt_1>_with_XML_REL_ACC					
Parameter: <REMS_notification_2>_for_Acceptance					
Parameter: <REMS_receipt_2>_with_XML_NOT_F_ACC					
Parameter: <REMS_receipt_3>_with_XML_CONS_ACC					
Parameter: <REMS_receipt_4>_with_XML_REL_ACC					
Parameter: <REMS_receipt_5>_EV_SET#1					
Parameter: <REMS_receipt_6>_XML_CONT_HAND					
Sequence of actions					
#	**Sender**	**SREMS**	**RREMS**	**Receiving side**	
1	Sender sends original message				
2		Accepts submission. Generates XML_SUB_ACC ERDS evidence			
3		Generates <REM_dispatch_1>_with_XML_SUB_ACC and stores it internally			
4		Relays <REMS_notification_1>_for_Acceptance			
5			Accepts <REMS_notification_1>_for_Acceptance		
6			Generates <REMS_notification_2>_for_Acceptance		
7			Sends <REMS_notification_2>_for_Acceptance to receiving side		
8			Generates XML_NOT_F_ACC ERDS evidence		
9			Generates <REMS_receipt_2>_with_XML_NOT_F_ACC		
10			Sends <REMS_receipt_2>_with_XML_NOT_F_ACC back to SREMS		
11		Receives <REMS_receipt_2>_with_XML_NOT_F_ACC			
12				All the entities at receiving side answer positively to SREMS	

Scenario id: SREMS_SN_RREMS_SF#4					Purpose
13		Generates XML_CONS_ACC ERDS evidence for all entities at receiving side			
14		Relays <REM_dispatch_1>_with_XML_SUB_ACC to RREMS			
15		Generates <REMS_receipt_3>_with_XML_CONS_ACC	Receives <REM_dispatch_1>_with_XML_SUB_ACC		
16		Sends <REMS_receipt_3>_with_XML_CONS_ACC back to sender	Generates XML_REL_ACC ERDS evidence		
17	Receives <REMS_receipt_3>_with_XML_CONS_ACC		Generates <REMS_receipt_4>_with_XML_REL_ACC ERDS evidence		
18			Sends it back to SREMS		
19		Receives <REMS_receipt_4>_with_XML_REL_ACC	Generates <REM_dispatch_2>_with_XML_SUB_ACC		
20			Consigns it to receiving side		
21				N-1 consignments succeed, one fails	
22			Generates XML_CONT_CONS for N-1 entities and one XML_CONS_FAIL for one entity		
23			Generates <REMS_receipt_5>_with_EV_SET#1		
24			Sends <REMS_receipt_5>_with_EV_SET#1 back to SREMS		
25		Receives <REMS_receipt_4>_with_EV_SET#1			
26					
27				N-1 entities at receiving side retrieve the user content	
28			Generates XML_CONT_HAND ERDS evidence for N-1 entities		
29			Generates <REMS_receipt_5>_with_XML_CONT_HAND		
30			Sends <REMS_receipt_5>_with_XML_CONT_HAND to SREMS		

Scenario id: SREMS_SN_RREMS_SF#4					Purpose
31		Receives <REMS_receipt_5>_with_XML_CONT_HAND			
32		Sends <REMS_receipt_5>_with_XML_CONT_HAND back to sender			
33	Receives <REMS_receipt_5>_with_XML_CONT_HAND				

Table 7d: Scenarios where SREMS operates Store&Notify and RREMSs operate Store&Forward (5/5)

Scenario id: SREMS_SN_RREMS_SF#5					Purpose
Parameter: <REM_dispatch_1>_with_XML_SUB_ACC			Var EV_SET#1 = {CONT_CONS, CONT_HAND, CONT_HAND_FAIL }		As the first scenario in the present table but now one of the handovers to the receiving side fails
Parameter: <REMS_notification_1>_for_Acceptance					
Parameter: <REMS_receipt_1>_with_XML_REL_ACC					
Parameter: <REMS_notification_2>_for_Acceptance					
Parameter: <REMS_receipt_2>_with_XML_NOT_F_ACC					
Parameter: <REMS_receipt_3>_with_XML_CONS_ACC					
Parameter: <REMS_receipt_4>_with_XML_REL_ACC					
Parameter: <REMS_receipt_5>_EV_SET#1					
Sequence of actions					
#	**Sender**	**SREMS**	**RREMS**	**Receiving side**	
1	Sender sends original message				
2		Accepts submission. Generates XML_SUB_ACC ERDS evidence			
3		Generates <REM_dispatch_1>_with_XML_SUB_ACC and stores it internally			
4		Relays <REMS_notification_1>_for_Acceptance			
5			Accepts <REMS_notification_1>_for_Acceptance		
6			Generates <REMS_notification_2>_for_Acceptance		
7			Sends <REMS_notification_2>_for_Acceptance to receiving side		
8			Generates XML_NOT_F_ACC ERDS evidence		

Scenario id: SREMS_SN_RREMS_SF#5					Purpose
9			Generates <REMS_receipt_2>_with_XML_NOT_F_ACC		
10			Sends <REMS_receipt_2>_with_XML_NOT_F_ACC back to SREMS		
11		Receives <REMS_receipt_2>_with_XML_NOT_F_ACC			
12				All the entities at receiving side answer positively to SREMS.	
13		Generates XML_CONS_ACC ERDS evidence for all entities at receiving side			
14		Relays <REM_dispatch_1>_with_XML_SUB_ACC to RREMS			
15		Generates <REMS_receipt_3>_with_XML_CONS_ACC ERDS evidence	Receives <REM_dispatch_1>_with_XML_SUB_ACC		
16		Sends <REMS_receipt_3>_with_XML_CONS_ACC to sender	Generates XML_REL_ACC ERDS evidence		
17	Receives <REMS_receipt_3>_with_XML_CONS_ACC		Generates <REMS_receipt_4>_with_XML_REL_ACC ERDS evidence		
18			Sends it back to SREMS		
19		Receives <REMS_receipt_4>_with_XML_REL_ACC	Generates <REM_dispatch_2>_with_XML_SUB_ACC		
20			Consigns it to receiving side		
21			Generates XML_CONT_CONS ERDS evidence for all entities at receiving side	N-1 entities at receiving side retrieve the user content. One entity fails when trying to retrieve.	
22			Generates XML_CONT_HAND ERDS evidence for N-1 entities and XML_CONT_HAND_FAIL for one entity		
23			Generates <REMS_receipt_5>_with_EV_SET#1		
24			Sends <REMS_receipt_5>_with_EV_SET#1 to SREMS		
25		Receives <REMS_receipt_5>_with_EV_SET#1			

Scenario id: SREMS_SN_RREMS_SF#5					**Purpose**
26		Sends <REMS_receipt_5>_with_EV_SET#1 back to sender			
27	Receives <REMS_receipt_5>_with_EV_SET#1				

5.5 Scenarios for extended model

5.5.1 Introduction

The present clause defines test cases for scenarios that take place when the sender and the entities at the receiving side are subscribed to different REMSs and there is one intermediate IREMS between the SREMS and the RREMSs.

Clause 5.5.2 defines test cases when the all the REMSs operate in Store and Forward style.

Clause 5.5.3 defines test cases when the SREMS and the RREMS operate in Store and Forward style and the IREMS operates in Store and Notify style.

5.5.2 Scenarios for S&F->S&F->S&F

Table 8 shows scenarios where SREMS, IREMS and RREMS all operate in Store and Forward style.

The sets of scenarios shown in Table 8 extend the set shown in clause 4.4.2.1 of ETSI EN 319 532-1 [3].

Table 8: Scenarios where SREMS, IREMS, RREMSs all operate in Store&Forward style (1/5)

Scenario id: SREMS_SF_IREMS_SF_RREMS_SF#1						**Purpose**
Parameter: <REM_dispatch_1>_with_XML_SUB_ACC						First scenario where the N entities successfully retrieve the REM dispatch with the user content
Parameter: <REMS_receipt_1>_with_XML_REL_ACC						
Parameter: <REMS_receipt_2>_with_XML_REL_ACC						
Parameter: <REMS_receipt_3>_with_XML_CONT_CONS						
Parameter: <REMS_receipt_4>_with_XML_CONT_HAND						
Sequence of actions						
#	**Sender**	**SREMS**	**IREMS**	**RREMS**	**Receiving side**	
1	Sender sends original message					
2		Accepts submission and generates XML_SUB_ACC ERDS evidence				
3		Generates <REM_dispatch_1>_with_XML_SUB_ACC				
4		Relays it to IREMS				
5			Receives <REM_dispatch_1>_with_XML_SUB_ACC			
6			Generates XML_REL_ACC ERDS evidence			
7			Generates <REMS_receipt_1>_with_XML_REL_ACC			
8			Sends it back to SREMS			
9		Receives <REMS_receipt_1>_with_XML_REL_ACC	Relays <REM_dispatch_1>_with_XML_SUB_ACC to RREMS			
10				Receives <REM_dispatch_1>_with_XML_SUB_ACC		
11				Generates XML_REL_ACC ERDS evidence		
12				Generates <REMS_receipt_2>_with_XML_REL_ACC		
13				Sends it back to IREMS		
14			Receives <REMS_receipt_2>_with_XML_REL_ACC	Consigns <REM_dispatch_1>_with_XML_SUB_ACC to receiving side		

Scenario id: SREMS_SF_IREMS_SF_RREMS_SF#1						Purpose
15					<REM_dispatch_1>_with_XML_SUB_ACC consigned to all the entities in receiving side	
16				Generates XML_CONT_CONS ERDS evidence		
17				Generates <REMS_receipt_3>_with_XML_CONT_CONS	All the entities in receiving side retrieve <REM_dispatch_1>_with_XML_SUB_ACC	
18				Sends it back to IREMS		
19			Receives <REMS_receipt_3>_with_XML_CONT_CONS	Generates XML_CONT_HAND ERDS evidence		
20			Sends it back to SREMS	Generates <REMS_receipt_4>_with_XML_CONT_HAND		
21		Receives <REMS_receipt_3>_with_XML_CONT_CONS		Sends it back to IREMS		
22		Sends it back to sender	Receives <REMS_receipt_4>_with_XML_CONT_HAND			
23	Receives <REMS_receipt_3>_with_XML_CONT_CONS		Sends it back to SREMS			
24		Receives <REMS_receipt_4>_with_XML_CONT_HAND				
25		Sends it back to sender				
26	Receives <REMS_receipt_4>_with_XML_CONT_HAND					

Table 8a: Scenarios where SREMS, IREMS, RREMSs all operate in Store&Forward style (2/5)

Scenario id: SREMS_SF_IREMS_SF_RREMS_SF#2						Purpose
Parameter: <REM_dispatch_1>_with_XML_SUB_ACC			Var EV_SET#1 = {CONT_CONS, CONT_CONS_FAIL}			As the first scenario but now one of the consignments fails
Parameter: <REMS_receipt_1>_with_XML_REL_ACC						
Parameter: <REMS_receipt_2>_with_XML_REL_ACC						
Parameter: <REMS_receipt_3>_with_EV_SET#1						
Parameter: <REMS_receipt_4>_with_XML_CONT_HAND						
Sequence of actions						
#	**Sender**	**SREMS**	**IREMS**	**RREMS**	**Receiving side**	
1	Sender sends original message					
2		Accepts submission and generates XML_SUB_ACC ERDS evidence				
3		Generates <REM_dispatch_1>_with_XML_SUB_ACC				
4		Relays it to IREMS				
5			Receives <REM_dispatch_1>_with_XML_SUB_ACC			
6			Generates XML_REL_ACC ERDS evidence			
7			Generates <REMS_receipt_1>_with_XML_REL_ACC			
8			Sends it back to SREMS			
9		Receives <REMS_receipt_1>_with_XML_REL_ACC	Relays <REM_dispatch_1>_with_XML_SUB_ACC to RREMS			
10				Receives <REM_dispatch_1>_with_XML_SUB_ACC		
11				Generates XML_REL_ACC ERDS evidence		
12				Generates <REMS_receipt_2>_with_XML_REL_ACC		
13				Sends it back to IREMS		
14			Receives <REMS_receipt_2>_with_XML_REL_ACC	Consigns <REM_dispatch_1>_with_XML_SUB_ACC to the receiving side		

Scenario id: SREMS_SF_IREMS_SF_RREMS_SF#2						Purpose
15					<REM_dispatch_2>_with_XML_SUB_ACC is consigned to N-1 entities in receiving side. One of the consignments fails	
16				Generates XML_CONT_CONSERDS evidence for N-1 entities and XML_CONT_CONS_FAILERDS evidence for one entity		
17				Generates <REMS_receipt_3>_with_EV_SET#1	N-1 entities in receiving side retrieve <REM_dispatch_2>_with_XML_SUB_ACC	
18				Sends it back to IREMS		
19			Receives <REMS_receipt_3>_with_EV_SET#1	Generates XML_CONT_HAND for N-1 entities		
20			Sends it back to SREMS	Generates <REMS_receipt_4>_with_XML_CONT_HAND		
21		Receives <REMS_receipt_3>_with_EV_SET#1		Sends it back to IREMS		
22		Sends it back to sender	Receives <REMS_receipt_4>_with_XML_CONT_HAND			
23	Receives <REMS_receipt_3>_with_EV_SET#1		Sends it back to SREMS			
24		Receives <REMS_receipt_4>_with_XML_CONT_HAND				
25		Sends it back to sender				
26	Receives <REMS_receipt_4>_with_XML_CONT_HAND					

Table 8b: Scenarios where SREMS, IREMS, RREMSs all operate in Store&Forward style (3/5)

Scenario id: SREMS_SF_IREMS_SF_RREMS_SF#3		Purpose
Parameter: <REM_dispatch_1>_with_XML_SUB_ACC	Var EV_SET#1 = { CONT_HAND, CONT_HAND_FAIL }	As the first scenario but now one of the entities fails when trying to retrieve the user content
Parameter: <REMS_receipt_1>_with_XML_REL_ACC		
Parameter: <REMS_receipt_2>_with_XML_REL_ACC		
Parameter: <REMS_receipt_3>_with_XML_CONT_CONS		
Parameter: <REMS_receipt_4>_with_EV_SET#1		

Sequence of actions

#	Sender	SREMS	IREMS	RREMS	Receiving side
1	Sender sends original message				
2		Accepts submission and generates XML_SUB_ACC ERDS evidence			
3		Generates <REM_dispatch_1>_with_XML_SUB_ACC			
4		Relies it to IREMS			
5			Receives <REM_dispatch_1>_with_XML_SUB_ACC		
6			Generates XML_REL_ACC ERDS evidence		
7			Generates <REMS_receipt_1>_with_XML_REL_ACC		
8			Sends it back to SREMS		
9		Receives <REMS_receipt_1>_with_XML_REL_ACC	Relies <REM_dispatch_1>_with_XML_SUB_ACC to RREMS		
10				Receives <REM_dispatch_1>_with_XML_SUB_ACC to RREMS	
11				Generates XML_REL_ACC ERDS evidence	
12				Generates <REMS_receipt_2>_with_XML_SUB_ACC	
13				Sends it back to IREMS	
14			Receives <REMS_receipt_2>_with_XML_SUB_ACC	Generates <REM_dispatch_2>_with_XML_SUB_ACC to RREMS	
15				Consigns it to receiving side	

Scenario id: SREMS_SF_IREMS_SF_RREMS_SF#3						Purpose
16					<REM_dispatch_2>_with_XML_SUB_ACC to RREMS is consigned all the entities	
17				Generates XML_CONT_CONS ERDS evidence		
18				Generates <REMS_receipt_3>_with_XML_ XML_CONT_CONS	N-1 entities in receiving side retrieve <REM_dispatch_2>_with_XML_SUB_ACC to RREMS. One entity fails when trying to retrieve it	
19				Sends it back to IREMS		
20			Receives <REMS_receipt_3>_with_XML_ XML_CONT_CONS	Generates XML_CONT_HAND ERDS evidence for N-1 entities and XML_CONT_HAND_FAIL for one entity		
21			Sends it back to SREMS	Generates <REMS_receipt_4>_with_EV_SET#1		
22		Receives <REMS_receipt_3>_with_XML_ XML_CONT_CONS		Sends it back to IREMS		
23		Sends it back to sender	Receives <REMS_receipt_4>_with_EV_SET#1			
24	Receives <REMS_receipt_3>_with_XML_ XML_CONT_CONS		Sends it back to SREMS			
25		Receives <REMS_receipt_4>_with_EV_SET#1				
26		Sends it back to sender				
27	Receives <REMS_receipt_4>_with_EV_SET#1					

Table 8c: Scenarios where SREMS, IREMS, RREMSs all operate in Store&Forward style (4/5)

Scenario id: SREMS_SF_IREMS_SF_RREMS_SF#4						Purpose
Parameter: <REM_dispatch_1>_with_XML_SUB_ACC						As the first scenario but now the IREMS rejects relaying
Parameter: <REMS_receipt_1>_with_XML_REL_REJ						
Sequence of actions						
#	Sender	SREMS	IREMS	RREMS	Receiving side	
1	Sender sends original message					
2		Accepts submission and generates XML_SUB_ACC ERDS evidence				
3		Generates <REM_dispatch_1>_with_XML_SUB_ACC				
4		Relays it to IREMS				
5			Rejects relay			
6			Generates XML_REL_REJ ERDS evidence			
7			Generates <REMS_receipt_1>_with_XML_REL_REJ			
8			Sends it back to SREMS			
9		Receives <REMS_receipt_1>_with_XML_REL_REJ				
10		Sends it back to sender				
11	Receives <REMS_receipt_1>_with_XML_REL_REJ					

Table 8d: Scenarios where SREMS, IREMS, RREMSs all operate in Store&Forward style (5/5)

Scenario id: SREMS_SF_IREMS_SF_RREMS_SF#5						Purpose
Parameter: <REM_dispatch_1>_with_XML_SUB_ACC						As the first scenario but now the RREMS rejects relaying
Parameter: <REMS_receipt_1>_with_XML_REL_ACC						
Parameter: <REMS_receipt_2>_with_XML_REL_REJ						
Sequence of actions						
#	Sender	SREMS	IREMS	RREMS	Receiving side	
1	Sender sends original message					
2		Accepts submission and generates XML_SUB_ACC ERDS evidence				

Scenario id: SREMS_SF_IREMS_SF_RREMS_SF#5						Purpose
3		Generates <REM_dispatch_1>_with_XML_SUB_ACC				
4		Relays it to IREMS				
5			Receives <REM_dispatch_1>_with_XML_SUB_ACC			
6			Generates XML_REL_ACC ERDS evidence			
7			Generates <REMS_receipt_1>_with_XML_REL_ACC			
8			Sends it back to SREMS			
9		Receives <REMS_receipt_1>_with_XML_REL_ACC	Relays <REM_dispatch_1>_with_XML_SUB_ACC to RREMS			
10				Receives <REM_dispatch_1>_with_XML_SUB_ACC but it rejects it		
11				Generates XML_REL_REJ ERDS evidence		
12				Generates <REMS_receipt_2>_with_XML_REL_REJ		
13				Sends it back to IREMS		
14			Receives <REMS_receipt_2>_with_XML_REL_REJ			
15			Sends it back to SREMS			
16		Receives <REMS_receipt_2>_with_XML_REL_REJ				
17		Sends it back to sender				
18	Receives <REMS_receipt_2>_with_XML_REL_REJ					

5.5.3 Scenarios for S&F -> S&N -> S&F

Table 9 shows scenarios where SREMS, and RREMS operate Store and Forward style and IREMS operates Store and Notify.

The sets of scenarios shown in Table 9 extend the set shown in clause 4.4.2.2 of ETSI EN 319 532-1 [3].

Table 9: Scenarios where SREMS and RREMSs operate Store&Forward style and IREMS operates Store&Notify (1/3)

Scenario id: SREMS_SF_IREMS_SN_RREMS_SF#1						**Purpose**
Parameter: <REM_dispatch_1>_with_XML_SUB_ACC						First scenario where all the entities at receiving side successfully retrieve the user content
Parameter: <REMS_receipt_1>_with_XML_REL_ACC						
Parameter: <REMS_notification_1>_for_Acceptance						
Parameter: <REMS_receipt_2>_with_XML_NOT_F_ACC						
Parameter: <REMS_receipt_3>_with_XML_CONS_ACC						
Parameter: <REMS_receipt_4>_with_XML_REL_ACC						
Parameter: <REM_dispatch_2>_with_XML_SUB_ACC						
Parameter: <REMS_receipt_5>_with_XML_CONT_CONS						
Parameter: <REMS_receipt_6>_with_XML_CONT_HAND						
Sequence of actions						
#	**Sender**	**SREMS**	**IREMS**	**RREMS**	**Receiving side**	
1	Sender sends original message					
2		Accepts submission and generates XML_SUB_ACC ERDS evidence				
3		Generates <REM_dispatch_1>_with_XML_SUB_ACC				
4		Relays it to IREMS				
5			Receives <REM_dispatch_1>_with_XML_SUB_ACC and stores it			
6			Generates XML_REL_ACC ERDS evidence			
7			Generates <REMS_receipt_1>_with_XML_REL_ACC			
8			Sends it back to SREMS			
9		Receives <REMS_receipt_1>_with_XML_REL_ACC	Generates <REMS_notification_1>_for_Acceptance			
10			Relays <REMS_notification_1>_for_Acceptance to RREMS			
11				Receives <REMS_notification_1>_for_Acceptance		
12				Sends <REMS_notification_1>_for_Acceptance to receiving side		

Scenario id: SREMS_SF_IREMS_SN_RREMS_SF#1						Purpose
13				Generates XML_NOT_F_ACC ERDS evidence	All the parties at receiving side receive <REMS_notification_1>_for _Acceptance	
14				Generates <REMS_receipt_2>_with_XML _NOT_F_ACC		
15				Sends it back to IREMS		
16			Receives <REMS_receipt_2>_with_XM L_NOT_F_ACC			
17			Sends it back to SREMS			
18		Receives <REMS_receipt_2>_with_X ML_NOT_F_ACC				
19					All the parties at receiving side access IREMS and accept consignment	
20			Generates XML_CONS_ACC for all the N entities at receiving side			
21			Generates <REMS_receipt_3>_with_XM L_CONS_ACC			
22			Sends it back to SREMS			
23		Receives <REMS_receipt_3>_with_X ML_CONS_ACC	Relays <REM_dispatch_1>_with_XM L_SUB_ACC to RREMS			
24				Accepts <REM_dispatch_1>_with_XML _SUB_ACC		
25				Generates XML_REL_ACC ERDS evidence		
26				Generates <REMS_receipt_4>_with_XML _REL_ACC		
27				Sends it back to IREMS		
28			Receives <REMS_receipt_4>_with_XM L_REL_ACC	Generates <REM_dispatch_2>_with_XML _SUB_ACC		
29				Consigns <REM_dispatch_2>_with_XML _SUB_ACC to receiving side		

Scenario id: SREMS_SF_IREMS_SN_RREMS_SF#1						Purpose
30					<REM_dispatch_2>_with_XML_REL_ACC is consigned to all entities in receiving side	
31				Generates XML_CONT_CONS ERDS evidence		
32				Generates <REMS_receipt_5>_with_XML_CONT_CONS		
33				Sends it back to IREMS	All entities in receiving side retrieve user content	
34			Receives <REMS_receipt_5>_with_XML_CONT_CONS	Generates XML_CONT_CONT_HAND evidence		
35			sends it back to SREMS	Generates <REMS_receipt_6>_with_XML_CONT_HAND		
36		Receives <REMS_receipt_5>_with_XML_CONT_CONS		Sends it back to IREMS		
37		Sends it back to sender	Receives <REMS_receipt_6>_with_XML_CONT_HAND			
38	Receives <REMS_receipt_5>_with_XML_CONT_CONS		sends it back to SREMS			
39		Receives <REMS_receipt_6>_with_XML_CONT_HAND				
40		Sends it back to sender				
41	Receives <REMS_receipt_6>_with_XML_CONT_HAND					

Table 9a: Scenarios where SREMS and RREMSs operate Store&Forward style and IREMS operates Store&Notify (2/3)

Scenario id: SREMS_SF_IREMS_SN_RREMS_SF#2						**Purpose**
Parameter: <REM_dispatch_1>_with_XML_SUB_ACC			Var EV_SET#1 = {XML_CONS_ACC, XML_CONS_REJ}			As first scenario but now one of the entities at receiving side rejects consignment
Parameter: <REMS_receipt_1>_with_XML_REL_ACC						
Parameter: <REMS_notification_1>_for_Acceptance						
Parameter: <REMS_receipt_2>_with_XML_NOT_F_ACC						
Parameter: <REMS_receipt_3>_with_EV_SET#1						
Parameter: <REMS_receipt_4>_with_XML_REL_ACC						
Parameter: <REM_dispatch_2>_with_XML_SUB_ACC						
Parameter: <REMS_receipt_5>_with_XML_CONT_CONS						
Parameter: <REMS_receipt_6>_with_XML_CONT_HAND						
Sequence of actions						
#	**Sender**	**SREMS**	**IREMS**	**RREMS**	**Receiving side**	
1	Sender sends original message					
2		Accepts submission and generates XML_SUB_ACC ERDS evidence				
3		Generates <REM_dispatch_1>_with_XML_SUB_ACC				
4		Relays it to IREMS				
5			Receives <REM_dispatch_1>_with_XML_SUB_ACC and stores it			
6			Generates XML_REL_ACC ERDS evidence			
7			Generates <REMS_receipt_1>_with_XML_REL_ACC			
8			Sends it back to SREMS			
9		Receives <REMS_receipt_1>_with_XML_REL_ACC	Generates <REMS_notification_1>_for_Acceptance			
10			Relays <REMS_notification_1>_for_Acceptance to RREMS			
11				Receives <REMS_notification_1>_for_Acceptance		
12				Sends <REMS_notification_1>_for_Acceptance to receiving side		

Scenario id: SREMS_SF_IREMS_SN_RREMS_SF#2						Purpose
13				Generates XML_NOT_F_ACC ERDS evidence	All the parties at receiving side receive <REMS_notification_1>_for _Acceptance	
14				Generates <REMS_receipt_2>_with_XML _NOT_F_ACC	.	
15				Sends it back to IREMS		
16			Receives <REMS_receipt_2>_with_XM L_NOT_F_ACC			
17			Sends it back to SREMS		N-1 parties at receiving side access IREMS and accept consignment. One party rejects it	
18		Receives <REMS_receipt_2>_with_X ML_NOT_F_ACC	Generates XML_CONS_ACC ERDS evidence for N-1 entities at receiving side and one XML_CONS_REJ ERDS evidence for one entity at receiving side.			
19			Generates <REMS_receipt_3>_with_EV _SET#1			
20			Sends it back to SREMS			
21		Receives <REMS_receipt_3>_ with_EV_SET#1	Relays <REM_dispatch_1>_with_XM L_SUB_ACC to RREMS for the N-1 accepting entities			
22		Sends it back to sender		Accepts <REM_dispatch_1>_with_XML _SUB_ACC		
23	Receives <REMS_receipt_3>_ with_EV_SET#1			Generates XML_REL_ACC ERDS evidence		
24				Generates <REMS_receipt_4>_with_XML _REL_ACC		
25				Sends it back to IREMS		
26			Receives <REMS_receipt_4>_with_XM L_REL_ACC	Generates <REM_dispatch_2>_with_XML _SUB_ACC		
27				Consigns <REM_dispatch_2>_with_XML _SUB_ACC to N-1 accepting entities at receiving side		

Scenario id: SREMS_SF_IREMS_SN_RREMS_SF#2						Purpose
28					<REM_dispatch_2>_with_XML_SUB_ACC is consigned to N-1 entities at receiving side	
29				Generates XML_CONT_CONS ERDS evidence for N-1 entities		
30				Generates <REMS_receipt_5>_with_XML_CONT_CONS		
31				Sends it back to IREMS	N-1 entities in receiving side retrieve user content	
32			Receives <REMS_receipt_5>_with_XML_CONT_CONS	Generates XML_CONT_HAND evidence for N-1 entities		
33			sends it back to SREMS	Generates <REMS_receipt_6>_with_XML_CONT_HAND		
34		Receives <REMS_receipt_5>_with_XML_CONT_CONS		Sends it back to IREMS		
35		Sends it back to sender	Receives <REMS_receipt_6>_with_XML_CONT_HAND			
36	Receives <REMS_receipt_5>_with_XML_CONT_CONS		sends it back to SREMS			
37		Receives <REMS_receipt_6>_with_XML_CONT_HAND				
38		Sends it back to sender				
39	Receives <REMS_receipt_6>_with_XML_CONT_HAND					

Table 9b: Scenarios where SREMS and RREMSs operate Store&Forward style and IREMS operates Store&Notify (3/3)

Scenario id: SREMS_SF_IREMS_SN_RREMS_SF#3		**Purpose**
Parameter: <REM_dispatch_1>_with_XML_SUB_ACC	Var EV_SET#1 = {XML_CONS_ACC, XML_CONS_REJ}	As first scenario but now one of the entities at receiving side rejects consignment and another entity fails in retrieving the user conten
Parameter: <REMS_receipt_1>_with_XML_REL_ACC	Var EV_SET#2 = {XML_CONT_HAND, XML_CONT_HAND_FAIL}	
Parameter: <REMS_notification_1>_for_Acceptance		
Parameter: <REMS_receipt_2>_with_XML_NOT_F_ACC		
Parameter: <REMS_receipt_3>_with_EV_SET#1		
Parameter: <REMS_receipt_4>_with_XML_REL_ACC		
Parameter: <REM_dispatch_2>_with_XML_SUB_ACC		
Parameter: <REMS_receipt_5>_with_XML_CONT_CONS		
Parameter: <REMS_receipt_6>_with_EV_SET#2		

Sequence of actions

#	Sender	SREMS	IREMS	RREMS	Receiving side
1	Sender sends original message				
2		Accepts submission and generates XML_SUB_ACC ERDS evidence			
3		Generates <REM_dispatch_1>_with_XML_SUB_ACC			
4		Relays it to IREMS			
5			Receives <REM_dispatch_1>_with_XML_SUB_ACC and stores it		
6			Generates XML_REL_ACC ERDS evidence		
7			Generates <REMS_receipt_1>_with_XML_REL_ACC		
8			Sends it back to SREMS		
9		Receives <REMS_receipt_1>_with_XML_REL_ACC	Generates <REMS_notification_1>_for_Acceptance		
10			Relays <REMS_notification_1>_for_Acceptance to RREMS		
11				Receives <REMS_notification_1>_for_Acceptance	
12				Sends <REMS_notification_1>_for_Acceptance to receiving side	

Scenario id: SREMS_SF_IREMS_SN_RREMS_SF#3						**Purpose**
13				Generates XML_NOT_F_ACC ERDS evidence	All the parties at receiving side receive <REMS_notification_1>_for _Acceptance	
14				Generates <REMS_receipt_2>_with_XML _NOT_F_ACC	.	
15				Sends it back to IREMS		
16			Receives <REMS_receipt_2>_with_XM L_NOT_F_ACC			
17			Sends it back to SREMS		N-1 parties at receiving side access IREMS and accept consignment. One party rejects it	
18		Receives <REMS_receipt_2>_with_X ML_NOT_F_ACC	Generates XML_CONS_ACC ERDS evidence for N-1 entities at receiving side and one XML_CONS_REJ ERDS evidence for one entity at receiving side.			
19			Generates <REMS_receipt_3>_with_EV _SET#1			
20			Sends it back to SREMS			
21		Receives <REMS_receipt_3>_with_E V_SET#1	Relays <REM_dispatch_1>_with_XM L_SUB_ACC to RREMS for the N-1 accepting entities			
22		Sends it back to sender		Accepts <REM_dispatch_1>_with_XML _SUB_ACC		
23	Receives <REMS_receipt_3>_ with_EV_SET#1			Generates XML_REL_ACC ERDS evidence		
24				Generates <REMS_receipt_4>_with_XML _REL_ACC		
25				Sends it back to IREMS		
26				Generates <REM_dispatch_2>_with_XML _SUB_ACC		
27			Receives <REMS_receipt_4>_with_XM L_REL_ACC	Consigns <REM_dispatch_2>_with_XML _SUB_ACC to N-1 accepting entities at receiving side		

Scenario id: SREMS_SF_IREMS_SN_RREMS_SF#3						Purpose
28					<REM_dispatch_2>_with_XML_SUB_ACC is consigned to N-1 entities at receiving side	
29				Generates XML_CONT_CONSERDS evidence for N-1 entities		
30				Generates <REMS_receipt_5>_with_XML_CONT_CONS		
31				Sends it back to IREMS	N-2 entities in receiving side retrieve user content, one entity fails	
32			Receives <REMS_receipt_5>_with_XML_CONT_CONS	Generates XML_CONT_HAND evidence for N-2 entities and XML_CONT_HAND_FAIL for one entity		
33			sends it back to SREMS	Generates <REMS_receipt_6>_with_EV_SET#2		
34		Receives <REMS_receipt_5>_with_XML_CONT_CONS		Sends it back to IREMS		
35		Sends it back to sender	Receives <REMS_receipt_6>_with_EV_SET#2			
36	Receives <REMS_receipt_5>_with_XML_CONT_CONS		sends it back to SREMS			
37		Receives <REMS_receipt_6>_with_EV_SET#2				
38		Sends it back to sender				
39	Receives <REMS_receipt_6>_with_EV_SET#2					

6 REM Messages instances

6.1 Introduction and technical approach

The present clause defines a number of instances of the different types of REM Messages, namely: REMS notification, REM payload, REMS receipt, and REM dispatch, as defined in ETSI EN 319 532-1 [3]. These instances are used in clause 8 for defining different test cases.

The set of REM message instances is built following the technical approach shown below:

- The set includes instances of each type of REM message.
- For each type of REM message there is at least one instance where all the optional headers defined in ETSI EN 319 532-3 [5] are present. In the rest of test cases subsets of the aforementioned optional headers are present.
- For each type of REM message there will be at least one instance where the REM message includes all the optional MIME parts that it can include (in addition to the mandatory ones). In the rest of instances, subsets of the aforementioned optional MIME parts are present.
- The present document first defines different instances for each MIME part that can be present in one of the different types of REM message, namely: REM message outermost header (that includes REMS relay metadata and REMS handover metadata), introduction MIME section, original message MIME section, extensions MIME section, ERDS evidence MIME section, and S/MIME signature MIME section.
- Each instance of a certain type of REM message is defined as a composition of different MIME parts specified in the aforementioned MIME parts test cases.

The rest of the present clause is organized as follows:

Clause 6.2 presents a number of combinations of fields for the different headers that can be present in the different REM message types. These combinations are specified separately as they are used in the definition of instances of different REM message types.

Clause 6.3 defines instances of REM payloads.

Clause 6.4 defines instances of REMS notifications.

Clause 6.5 defines instances of REMS receipts.

Clause 6.6 defines instances of REM dispatches.

As mentioned in clause 4.2 new combinations of fields may be defined for each header, and new instances of REM messages may be added to the current set, for defining new test cases.

6.2 Combinations of fields for headers in REM envelopes

6.2.1 Introduction

The following clauses define combinations of headers fields for all the headers that may be present in REM envelopes, namely: the outermost MIME header, the signed data MIME header, the headers in REMS introduction MIME section, the original message MIME section header, the REMS extensions MIME section header, the ERDS evidence MIME section header, and the REMS signature MIME header.

6.2.2 Combinations of fields for the outermost MIME header

The present clause defines different combinations of fields for the REM message outermost header of the REM envelope.

The definition of a certain combination is split in two tables, namely Table 10 and Table 11.

Table 10 shows combinations of fields that are defined in MIME and SMIME RFCs. Only the fields listed in a certain combination shall be present in that combination. Not listed fields shall be absent.

Cells in column "Field name" contain the name of the header fields.

Cells in column "Value" shows the value to be assigned to the header field. These cells may contain the following values:

- As specified in ETSI EN 319 532-3 [5]. This value is reserved for cases where the ETSI EN 319 532-3 [5] specifies a mandatory value for the header field.
- As recommended in ETSI EN 319 532-3 [5]. This value is reserved for cases where the ETSI EN 319 532-3 [5] recommends one certain value for the header field (usually using the modal verb should). In these cases, the header field has this recommended value in the combination.
- AS_PER_TESTCASE means that for this test case the tester is free to give to the aforementioned field the value it considers worth, provided that it fulfils the additional requirements indicated in Table 10.

Cells in "Notes/Additional requirements" contain one or more letters or/and one or more integer numbers. The letters correspond to additional requirements that are given after Table 10. The numbers correspond to numbers of notes that appear after the aforementioned additional requirements.

Table 10: Combinations of header fields defined in MIME and S/MIME RFCs

Combination identifier	Field name	Value	Notes/Additional requirements
RFCFields#1	Content-Type	As specified in ETSI EN 319 532-3 [5]	
	MIME-Version	As specified in ETSI EN 319 532-3 [5]	
	Message-ID	As recommended in ETSI EN 319 532-3 [5]	
	Date	As specified in ETSI EN 319 532-3 [5]	
	From	As recommended in ETSI EN 319 532-3 [5]	1
	To	As specified in ETSI EN 319 532-3 [5]	
	Subject	As recommended in ETSI EN 319 532-3 [5]	
RFCFields#2	Content-Type	As specified in ETSI EN 319 532-3 [5]	
	MIME-Version	As specified in ETSI EN 319 532-3 [5]	
	Message-ID	As recommended in ETSI EN 319 532-3 [5]	
	Date	As specified in ETSI EN 319 532-3 [5]	
	From	As recommended in ETSI EN 319 532-3 [5]	1
	To	As specified in ETSI EN 319 532-3 [5]	
	Cc	As recommended in ETSI EN 319 532-3 [5]	
	Subject	As recommended in ETSI EN 319 532-3 [5]	
	Reply-To	As specified in ETSI EN 319 532-3 [5]	
	Return-Path	As specified in ETSI EN 319 532-3 [5]	
	Received	As specified in ETSI EN 319 532-3 [5]	
	In-Reply-To	As recommended in ETSI EN 319 532-3 [5]	

NOTE 1: "From" header field is mandatory but ETSI EN 319 532-3 [5] recommends two ways of computing its value. The corresponding test cases for REM message formats will be in charge of selecting one of them. As it is expected that each REMS opts for one of the two mechanisms, and selecting one or the other does not introduce any hinder to interoperability, selecting one or the other does not result in a different test case.

Table 11 shows the combinations of the new header fields defined in ETSI EN 319 532-3 [5].

For the purpose of defining the test cases, the field "REM-ApplicablePolicy" shall always consist in a single URI and consequently, its formatting does not require creation of one MIME extension section as specified in ETSI EN 319 532-3 [5], clause 6.2.5. New combinations may be added where this field consists in a sequence of URIs and its structured value be placed in the corresponding extension MIME section.

Cells in "Header field name" column contain the name of a field in the REM message outermost header. The names used are the ones defined in ETSI EN 319 532-3 [5] and (whenever required) in the different RFCs specifying MIME and S/MIME formats (references [7], [8], [9], [10], [11], [12], [13] and [14]).

Cells in "Header field value" column contain either:

1) the value of the header field whose name is the one indicated in the previous column; or

2) AS_PER_TESTCASE, whit the meaning described before.

Cells in "Notes/Additional requirements" contain one or more letters or/and one or more integer numbers. The letters correspond to additional requirements that are given after the table. The numbers correspond to numbers of notes that appear after the aforementioned additional requirements.

Cells in "Purpose" contain a description of the purpose of the combination defined in the row.

Some rows of the table though only have three columns. This happens when a certain test case is based in a test case already specified and that has only some few differences. Then only the columns "Test case identifier", "Test case based on", and "Purpose" apply for these rows. The "Test case based on" central cell contains all the relevant details for the test case, including the test case on which this one is based and the differences between both of them.

The values of these fields shall be the ones specified in the aforementioned references.

Table 11 defines parameterized combinations for new headers defined in ETSI EN 319 532-3 [5]. The combinations have as parameters the assurance levels and the consignment mode.

Table 11: First set of parameterized combinations for REM message outermost header

Combination identifier	Header field name	Header field value	Notes/Additional requirements	Purpose
NewFields#1	REM-RelayDate	AS_PER_TESTCASE	a, 2	Use in tests where: the contents of the fields are correct; there is no indication neither of assurance levels nor of mode of consignmen
	REM-ExpirationDate	AS_PER_TESTCASE	b	
	REM-ScheduledDelivery	AS_PER_TESTCASE	c	
	REM-ApplicablePolicy	AS_PER_TESTCASE		
NewFields#2	REM-RelayDate	AS_PER_TESTCASE	a, 2	Use in tests where: the contents of the fields are correct; the assurance levels required is one parameter, and there is no indication of mode of consignment
	REM-ExpirationDate	AS_PER_TESTCASE	b	
	REM-RecipientAssuranceLevel	Parameter		
	REM-ScheduledDelivery	AS_PER_TESTCASE	c	
	REM-ApplicablePolicy	AS_PER_TESTCASE		
NewFields#3	REM-RelayDate	AS_PER_TESTCASE	a, 2	Use in tests where: the contents of the fields are correct; the outermost header has all the optional fields present; and the assurance levels and the mode of consignment are parameters
	REM-ExpirationDate	AS_PER_TESTCASE	b	
	REM-RecipientAssuranceLevel	Parameter		
	REM-ModeOfConsignment	Parameter		
	REM-ScheduledDelivery	AS_PER_TESTCASE	c	
	REM-ApplicablePolicy	AS_PER_TESTCASE		
NewFields#4	REM-RelayDate	AS_PER_TESTCASE	a, 2	Use in negative tests where: the scheduled delivery is after the expiration date; assurance levels is one parameter, and there is no indication of mode of consignment
	REM-ExpirationDate	AS_PER_TESTCASE	b	
	REM-RecipientAssuranceLeve	Parameter		
	REM-ScheduledDelivery	AS_PER_TESTCASE	d	
	REM-ApplicablePolicy	AS_PER_TESTCASE		
NewFields#5	REM-RelayDate	AS_PER_TESTCASE	a, 2	Use in negative tests where: the scheduled delivery is after the expiration date; and the assurance levels and the mode of consignment are parameters
	REM-ExpirationDate	AS_PER_TESTCASE	b	
	REM-RecipientAssuranceLevel	Parameter		
	REM-ModeOfConsignment	Parameter		
	REM-ScheduledDelivery	AS_PER_TESTCASE	d	
	REM-ApplicablePolicy	AS_PER_TESTCASE		
NewFields#6	REM-RelayDate	AS_PER_TESTCASE	a, 2	As NewFields#1 but without REM-ScheduledDelivery
	REM-ExpirationDate	AS_PER_TESTCASE		
	REM-ApplicablePolicy	AS_PER_TESTCASE		
NewFields#7	REM-RelayDate	AS_PER_TESTCASE	a, 2	As NewFields#2 but without REM-ScheduledDelivery
	REM-ExpirationDate	AS_PER_TESTCASE		
	REM-RecipientAssuranceLevel	Parameter		
	REM-ApplicablePolicy	AS_PER_TESTCASE		
NewFields#8	REM-RelayDate	AS_PER_TESTCASE	a, 2	As NewFields#3 but without REM-ScheduledDelivery
	REM-ExpirationDate	AS_PER_TESTCASE		
	REM-RecipientAssuranceLevel	Parameter		
	REM-ModeOfConsignment	Parameter		
	REM-ApplicablePolicy	AS_PER_TESTCASE		

Combination identifier	Header field name	Header field value	Notes/Additional requirements	Purpose
NewFields#9	REM-ExpirationDate	AS_PER_TESTCASE	b	As NewFields#1 but without REM-RelayDate and without REM-ScheduledDelivery
	REM-ApplicablePolicy	AS_PER_TESTCASE		
NewFields#10	REM-ExpirationDate	AS_PER_TESTCASE		As NewFields#2 but without REM-RelayDate and without REM-ScheduledDelivery
	REM-RecipientAssuranceLevel	Parameter		
	REM-ApplicablePolicy	AS_PER_TESTCASE		
NewFields#11	REM-ExpirationDate	AS_PER_TESTCASE		As NewFields#3 but without REM-RelayDate and without REM-ScheduledDelivery
	REM-RecipientAssuranceLevel	Parameter		
	REM-ModeOfConsignment	Parameter		
	REM-ApplicablePolicy	AS_PER_TESTCASE		
Additional requirements: a) The date and time indicated in this field shall be earlier than the date and time indicated in "REM-ExpirationDate" and "REM-ScheduledDelivery" (if present). b) The date and time indicated in this field shall be later than the date and time indicated in "REM-RelayDate" and "REM-ScheduledDelivery" (if present). c) The date and time indicated in this field shall be earlier than the date and time indicated in "REM-ScheduledDelivery" and later than the date and time indicated in "REM-RelayDate" (if present). d) The date and time indicated in this field shall be later than the date and time indicated in "REM-RelayDate" and later than "REM-ExpirationDate" (if present).				

NOTE 2: This combination can be used only in scenarios of REM messages where a REMS relays the REM message to another REMS, i.e. in situations where sender and receiver are not subscribed to the same REMS.

The present document defines the combinations of fields of the outermost header of the REM envelope that are shown in Table 12.

Each instance of the outermost header shown in Table 12 is defined as the aggregation of one of the combinations of RFC header fields defined in Table 10 and one of the combinations of the new header fields defined in Table 11.

Each combination defined in Table 12 has four parameters, namely:

1) An integer identifying the RFC headers combination.

2) An integer identifying the new header fields combinations.

3) The value of the required assurance levels (nil if this information is not present in the combination).

4) The consignment mode (nil if this information is not present).

The table defines a number of possible combinations of these parameters when the required assurance levels are the same for sender and entities at receiving side. It also defines combinations where the consignment mode field is not present. Finally, it also defines two illegal combinations, that can be used in negative test cases.

All the outermost headers instances can be obtained from the following expression:

OutermostHeader(RFCFields_id, NewFields_id,AssuranceLevelCombs, ConsignmentMode) where:

- RFCFields_id goes from 1 to 2.
- NewFields_id goes from 1 to 11.
- AssuranceLevelCombs is one of {nil, low/low, subs/subs, high/high}.
- ConsignmentMode is one of {nil, basic, consented, consentedSigned}.

Table 12 defines some combinations and their corresponding purposes.

Table 12: REM messages outermost header combinations

Combination identifier	Combination definition	Purpose
For test cases without REM-RelayDate and without REM-ScheduledDelivery. Scenarios without message relaying		
OutermostHeader(1,9,nil,nil)	RFCFields#1 + NewFields#9	No indications neither of assurance levels nor consignment mode
OutermostHeader(1,10,low/low,nil)	RFCFields#1 + NewFields#10(low/low)	No consignment mode indicated and assurance level
OutermostHeader(1,10,sub/subs,nil)	RFCFields#1 + NewFields#10(substantial/substantial)	
OutermostHeader(1,10,high/high,nil)	RFCFields#1 + NewFields#10(high/high)	
OutermostHeader (1,11,low/low,basic)	RFCFields#1 + NewFields#11(low/low, basic)	Basic consignment mode and assurance level
OutermostHeader(1,11,subs/subs,basic)	RFCFields#1 + NewFields#11(substantial/substantial, basic)	
OutermostHeader (1,11,high/high,basic)	RFCFields#1 + NewFields#11(high/high, basic)	
OutermostHeader (1,11,low/low,consented)	RFCFields#1 + NewFields#11(low/low, consented)	Consented consignment mode and assurance level
OutermostHeader(1,11,subs/subs,consented)	RFCFields#1 + NewFields#11(substantial/substantial, consented)	
OutermostHeader (1,11,high/high,consented)	RFCFields#1 + NewFields#11(high/high, consented)	
OutermostHeader (1,11,low/low,consentedSigned)	RFCFields#1 + NewFields#11(low/low, consentedSigned)	Consented and signed consignment mode and assurance level
OutermostHeader(1,11,subs/subs,consentedSigned)	RFCFields#1 + NewFields#11(substantial/substantial, consentedSigned)	
OutermostHeader (1,11,high/high,consentedSigned)	RFCFields#1 + NewFields#11(high/high, consentedSigned)	

Combination identifier	Combination definition	Purpose
For test cases without REM-ScheduledDelivery. Scenarios with message relaying		
OutermostHeader(1,6,nil,nil)	RFCFields#1 + NewFields#6	No indications neither of assurance levels nor consignment mode
OutermostHeader(1,7,low/low,nil)	RFCFields#1 + NewFields#7(low/low)	No consignment mode indicated and assurance level
OutermostHeader(1,7,sub/subs,nil)	RFCFields#1 + NewFields#7(substantial/substantial)	
OutermostHeader(1,7,high/high,nil)	RFCFields#1 + NewFields#7(high/high)	
OutermostHeader (1,8,low/low,basic)	RFCFields#1 + NewFields#8(low/low, basic)	Basic consignment mode and assurance level
OutermostHeader(1,8,subs/subs,basic)	RFCFields#1 + NewFields#8(substantial/substantial, basic)	
OutermostHeader (1,8,high/high,basic)	RFCFields#1 + NewFields#8(high/high, basic)	
OutermostHeader (1,8,low/low,consented)	RFCFields#1 + NewFields#8(low/low, consented)	Consented consignment mode and assurance level
OutermostHeader(1,8,subs/subs,consented)	RFCFields#1 + NewFields#8(substantial/substantial, consented)	
OutermostHeader (1,8,high/high,consented)	RFCFields#1 + NewFields#8(high/high, consented)	
OutermostHeader (1,8,low/low,consentedSigned)	RFCFields#1 + NewFields#8(low/low, consentedSigned)	Consented and signed consignment mode and assurance level
OutermostHeader(1,8,subs/subs,consentedSigned)	RFCFields#1 + NewFields#8(substantial/substantial, consentedSigned)	
OutermostHeader (1,8,high/high,consentedSigned)	RFCFields#1 + NewFields#8(high/high, consentedSigned)	
For test cases with REM-RelayDate and with REM-ScheduledDelivery. Scenarios with message relaying		
OutermostHeader(1,1,nil,nil)	RFCFields#1 + NewFields#1	No indications neither of assurance levels nor consignment mode
OutermostHeader(1,2,low/low,nil)	RFCFields#1 + NewFields#2(low/low)	No consignment mode indicated and assurance level
OutermostHeader(1,2,sub/subs,nil)	RFCFields#1 + NewFields#7(substantial/substantial)	
OutermostHeader(1,2,high/high,nil)	RFCFields#1 + NewFields#2(high/high)	
OutermostHeader (1,3,low/low,basic)	RFCFields#1 + NewFields#3(low/low, basic)	Basic consignment mode and assurance level
OutermostHeader(1,3,subs/subs,basic)	RFCFields#1 + NewFields#8(substantial/substantial, basic)	
OutermostHeader (1,3,high/high,basic)	RFCFields#1 + NewFields#3(high/high, basic)	
OutermostHeader (1,3,low/low,consented)	RFCFields#1 + NewFields#3(low/low, consented)	Consented consignment mode and assurance level
OutermostHeader(1,3,subs/subs,consented)	RFCFields#1 + NewFields#3(substantial/substantial, consented)	
OutermostHeader (1,3,high/high,consented)	RFCFields#1 + NewFields#3(high/high, consented)	
OutermostHeader (1,3,low/low,consentedSigned)	RFCFields#1 + NewFields#3(low/low, consentedSigned)	Consented and signed consignment mode and assurance level
OutermostHeader(1,3,subs/subs,consentedSigned)	RFCFields#1 + NewFields#3(substantial/substantial, consentedSigned)	
OutermostHeader (1,3,high/high,consentedSigned)	RFCFields#1 + NewFields#3(high/high, consentedSigned)	
OutermostHeader (1,4,low/low,nil)	RFCFields#1 + NewFields#4(low/low)	Combinations for negative test cases (cause of submission rejection for instance)
OutermostHeader (1,5,low/low,basic)	RFCFields#1 + NewFields#5(low/low, basic)	

Outer most headers combinations similar to the ones shown in the table but replacing 1 by 2 in the first parameter would result in combinations with RFCFields#2.

6.2.3 Combinations of fields for the signed data MIME header

In all the test cases defined in the present document this header shall be as specified in ETSI EN 319 532-3 [5], clause 6.2.2. The resulting combination of header fields will be identified as SIGDATA_COMB.

6.2.4 Combinations of fields for the REMS introduction section

6.2.4.1 Introduction

The present clause defines combinations of fields for three different headers in the REMS introduction section, namely: the REMS introduction MIME header, the multipart/alternative free text subsection header, and the multipart/alternative html subsection header.

6.2.4.2 Combinations of fields for the REMS introduction MIME header

All the test cases defined in the present shall include the "REM-Section-Type" field in this header. Its value shall be the recommended value in ETSI EN 319 532-3 [5], namely "rem_message/introduction". This combination is identified as REMS_IntrComb.

The mandatory field "REM-Content-Type" shall also be present with its value as specified in ETSI EN 319 532-3 [5].

6.2.4.3 Combinations of fields for the multipart/alternative free text subsection header

Table 13 shows the two possible combinations of fields for the multipart/alternative free text subsection header derived from ETSI EN 319 532-3 [5] specifications, namely one with the "Content-Disposition" field present and one without the "Content-Disposition" field.

Columns in Table 13 are identical to columns in Table 11. Values in "Header field value" have the same semantics as the values in column "Value" in Table 10.

Table 13: Combinations for the multipart/alternative free text subsection header

Combination identifier	Header field name	Header field value	Notes/Additional requirements	Purpose
FreeText#1	Content-Type	as specified in ETSI EN 319 532-3 [5]		Combination with Content-Disposition optional field present
	Content-Disposition	as specified in ETSI EN 319 532-3 [5]		
	Content-Transfer-Encoding	as recommended in ETSI EN 319 532-3 [5]		
FreeText#2	Content-Type	as specified in ETSI EN 319 532-3 [5]		Combination without Content-Disposition optional field
	Content-Transfer-Encoding	as recommended in ETSI EN 319 532-3 [5]		

6.2.4.4 Combinations of fields for the multipart/alternative html subsection header

All the test cases defined in the present document shall include the two mandatory header fields ("Content-Type" and "Content-Transfer-Encoding") in the multipart/alternative html subsection header, with the values specified in ETSI EN 319 532-3 [5]. This combination is identified as HTMLComb.

6.2.5 Combinations of fields for the original message MIME section header

Table 14 shows three combinations of fields for the original message section header derived from ETSI EN 319 532-3 [5] specifications.

Columns in Table 14 are identical to columns in Table 11. Values in "Header field value" have the same semantics as the values in column "Value" in Table 10 and the values in column "Value" in Table 11.

Table 14: Combinations for the original message MIME section header

Combination identifier	Header field name	Header field value	Notes/Additional requirements	Purpose
OrMess#1	Content-Type	as specified in ETSI EN 319 532-3 [5]		Combination with all the optional fields present
	Content-Disposition	as specified in ETSI EN 319 532-3 [5]		
	Content-Transfer-Encoding	as specified in ETSI EN 319 532-3 [5]		
	Content-Description	AS_PER_TESTCASE		
	REM-Section-Type	as recommended in ETSI EN 319 532-3 [5]		
OrMess#2	Content-Type	as specified in ETSI EN 319 532-3 [5]		Combination without Content-Description optional field
	Content-Disposition	as specified in ETSI EN 319 532-3 [5]		
	Content-Transfer-Encoding	as specified in ETSI EN 319 532-3 [5]		
	REM-Section-Type	as recommended in ETSI EN 319 532-3 [5]		
OrMess#3	Content-Type	as specified in ETSI EN 319 532-3 [5]		Combination without any optional field
	Content-Disposition	as specified in ETSI EN 319 532-3 [5]		
	Content-Transfer-Encoding	as specified in ETSI EN 319 532-3 [5]		

6.2.6 Combinations of fields for one REMS extension MIME section header

Table 15 shows three combinations of fields for one REMS extension MIME section header derived from ETSI EN 319 532-3 [5] specifications.

Columns in Table 15 are identical to columns in Table 11. Values in "Header field value" have the same semantics as the values in column "Value" in Table 10 and the values in column "Value" in Table 11.

Table 15: Combinations for one REMS extension MIME section header

Combination identifier	Header field name	Header field value	Notes/Additional requirements	Purpose
Ext#1	Content-Type	as recommended in ETSI EN 319 532-3 [5]		Combination with all the optional fields present
	Content-Disposition	as specified in ETSI EN 319 532-3 [5]		
	Content-Transfer-Encoding	as specified in ETSI EN 319 532-3 [5]		
	Content-Description	AS_PER_TESTCASE		
	REM-Section-Type	as recommended in ETSI EN 319 532-3 [5]		
	REM-Extension-Code	AS_PER_TESTCASE		
	REM-Extension-Namespace-URI	AS_PER_TESTCASE		
Ext#2	Content-Type	as specified in ETSI EN 319 532-3 [5]		Combination without REM-Extension-Code and REM-Extension-Namespace-URI optional fields
	Content-Disposition	as specified in ETSI EN 319 532-3 [5]		
	Content-Transfer-Encoding	as specified in ETSI EN 319 532-3 [5]		
	Content-Description	AS_PER_TESTCASE		
	REM-Section-Type	as recommended in ETSI EN 319 532-3 [5]		
Ext#3	Content-Type	as specified in ETSI EN 319 532-3 [5]		Combination without any optional field
	Content-Disposition	as specified in ETSI EN 319 532-3 [5]		
	Content-Transfer-Encoding	as specified in ETSI EN 319 532-3 [5]		

6.2.7 Combinations of fields for one ERDS evidence MIME section header

6.2.7.1 Combinations of fields for one XML ERDS evidence MIME section header

Table 16 shows three combinations of fields for one ERDS evidence MIME section header derived from ETSI EN 319 532-3 [5] specifications when the ERDS evidence is in XML format.

Columns in Table 16 are identical to columns in Table 11. Values in "Header field value" have the same semantics as the values in column "Value" in Table 10 and the values in column "Value" in Table 11.

Table 16: Combinations for one XML ERDS evidence MIME section header

Combination identifier	Header field name	Header field value	Notes/Additional requirements	Purpose
EVID#1	Content-Type	as recommended in ETSI EN 319 532-3 [5]		Combination with all the optional fields present
	Content-Disposition	as specified in ETSI EN 319 532-3 [5]		
	Content-Transfer-Encoding	as specified in ETSI EN 319 532-3 [5]		
	Content-Description	AS_PER_TESTCASE		
	REM-Section-Type	as recommended in ETSI EN 319 532-3 [5]		
EVID#2	Content-Type	as specified in ETSI EN 319 532-3 [5]		Combination without Content-Description optional field
	Content-Disposition	as specified in ETSI EN 319 532-3 [5]		
	Content-Transfer-Encoding	as specified in ETSI EN 319 532-3 [5]		
	REM-Section-Type	as recommended in ETSI EN 319 532-3 [5]		
EVID#3	Content-Type	as specified in ETSI EN 319 532-3 [5]		Combination without any optional field
	Content-Disposition	as specified in ETSI EN 319 532-3 [5]		
	Content-Transfer-Encoding	as specified in ETSI EN 319 532-3 [5]		

6.2.7.2 Combinations of fields for one PDF ERDS evidence MIME section header

Table 17 shows three combinations of fields for one ERDS evidence MIME section header derived from ETSI EN 319 532-3 [5] specifications when the ERDS evidence is a PDF document.

Columns in Table 17 are identical to columns in Table 11. Values in "Header field value" have the same semantics as the values in column "Value" in Table 10 and the values in column "Value" in Table 11.

NOTE: The identifiers for these combinations are identical to the identifiers for combinations of fields in headers of ERDS evidence sections containing XML ERDS evidence. This does not introduce any ambiguity in the present document, as the clauses that define test cases assign to each test case a unique identifier that includes a component indicating whether the ERDS evidence present in a certain REMS notification, REMS receipt, or REM dispatch are XML or PDF ERDS evidence.

Table 17: Combinations for one PDF ERDS evidence MIME section header

Combination identifier	Header field name	Header field value	Notes/Additional requirements	Purpose
EVID#1	Content-Type	as recommended in ETSI EN 319 532-3 [5]		Combination with all the optional fields present
	Content-Disposition	as specified in ETSI EN 319 532-3 [5]		
	Content-Transfer-Encoding	as specified in ETSI EN 319 532-3 [5]		
	Content-Description	AS_PER_TESTCASE		
	REM-Section-Type	as recommended in ETSI EN 319 532-3 [5]		
EVID#2	Content-Type	as specified in ETSI EN 319 532-3 [5]		Combination without Content-Description optional field
	Content-Disposition	as specified in ETSI EN 319 532-3 [5]		
	Content-Transfer-Encoding	as specified in ETSI EN 319 532-3 [5]		
	REM-Section-Type	as recommended in ETSI EN 319 532-3 [5]		
EVID#3	Content-Type	as specified in ETSI EN 319 532-3 [5]		Combination without any optional field
	Content-Disposition	as specified in ETSI EN 319 532-3 [5]		
	Content-Transfer-Encoding	as specified in ETSI EN 319 532-3 [5]		

6.2.8 Combinations of fields for the REMS signature MIME section header

Table 18 shows combinations of fields for the REMS signature section header derived from ETSI EN 319 532-3 [5] specifications, namely one with the "Content-Description" field present and one without the "Content- Description" field.

Columns in Table 18 are identical to columns in Table 11. Values in "Header field value" have the same semantics as the values in column "Value" in Table 10. In addition to that, a value enclosed in "" represents a literal value required for the field being dealt with.

Table 18: Combinations for the multipart/alternative free text subsection header

Combination identifier	Header field name	Header field value	Notes/Additional requirements	Purpose
Sig#1	Content-Type	as specified in ETSI EN 319 532-3 [5]		Combination with all the optional fields present
	Content-Transfer-Encoding	as specified in ETSI EN 319 532-3 [5]		
	Content-Disposition	as specified in ETSI EN 319 532-3 [5]		
	Content-Description	"S/MIME Cryptographic Signature"		
Sig#2	Content-Type	as specified in ETSI EN 319 532-3 [5]		Combination without Content-Description optional field
	Content-Transfer-Encoding	as recommended in ETSI EN 319 532-3 [5]		
	Content-Disposition	as specified in ETSI EN 319 532-3 [5]		
Sig#3	Content-Type	as specified in ETSI EN 319 532-3 [5]		Combination without any optional field
	Content-Disposition	as specified in ETSI EN 319 532-3 [5]		

6.3 Instances of REM payload

The present clause defines instances of REM payloads for being used in test cases.

Cells in column "Instance" shows the specific instance of REM payload and all its parametrized components. Each instance is the aggregation of one of the instances of the outmost header defined in clause 6.2.2 and a list of MIME sections. Each section is the aggregation of a MIME header and a MIME body. The MIME headers will be instances of the MIME headers as specified in clauses 6.2.3 to 6.2.8. Each REM payload instance is composed by the following components:

- the outermost header;
- the REM Introduction MIME section;
- the Free Text MIME section;
- the HTML Text MIME section;
- the Original Message MIME section;
- the Signature MIME section; and
- the optional Extension MIME section.

Each cell in column "Instance" identifies several REM payload instances, which have certain aspects in common and some other aspects that are different. This is achieved as indicated below:

The content of a cell in column "Instance" has the following pattern:

> **"REM_payloadInst (OuterMostHeader(RFCFields#I,NewFields#J,<AssLevel>,<Consignment>),REMSIntrCom,FreeText#K,HTMLComb,OrMess#L,Sig#M)"**

Where I, J, K, L, and M are integers. Assigning a value to I one of the existing combinations for RFCFields is selected (for instance RFCFields#1). Assigning a value to J one of the existing combinations for NewFields is selected (for instance NewFields#2). Assigning a value to K one of the existing combinations for FreeText is selected (for instance FreeText#1). Assigning a value to L one of the existing combinations for original message MIME section header is selected (for instance OrMess#1). Finally, assigning value to M one of the existing combinations for Signature MIME section header is selected (for instance OrMess#1).

<AssLevel> provides information on the assurance levels combinations, and can take as value either:

- nil, indicating that the header fields for indicating assurance levels are absent; OR
- "AssLevelComb", indicating that this parameter can take several values, each one corresponding to a combination of assurance levels.

<Consignment> provides information on the consignment mode, and can take as value either:

- nil, indicating that the header fields for indicating consignment mode is absent; OR
- "ConsignmentModeId", indicating that this parameter can take several values, each one corresponding to one specific consignment mode.

The aforementioned pattern is followed by text providing information of the values that I, J, K, L, M, "AssLevelComb" (if present), and "ConsignmentModeId" (if present) can take. The aforementioned pattern with this additional piece of information identifies a certain number of REM payloads in one unique cell.

For instance, a cell containing the following text:

REM_payloadInst (OuterMostHeader(RFCFields#I,NewFields#J,nil,nil),REMSIntrCom,FreeText#K,OrMess#L,Sig#M) where **Where**: I is one of {1,2}, J is one of {1, 6,9}, K is one of {1,2}, L is one of {1,2,3} and M is one of {1,2,3}

Would identify all those REM payloads built when:

- RFCFields#I is one of { RFCFields#1, RFCFields#2};
- NewFields#J is one of {NewFields#1, NewFields#6, NewFields#9};
- FreeText#K is one of {FreeText#1, FreeText#2};
- OrMess#L is one of {OrMess#1, OrMess#2, OrMess#3}; and
- Sig#M is one of {Sig#1, Sig#2, Sig#3}.

Similarly, a cell containing the following text:

REM_payloadInst (OuterMostHeader(RFCFields#I,NewFields#J,AssLevelComb,ConsignmentModeId),REMSIntrCom,FreeText#K, HTMLComb,OrMess#L,Sig#M) **Where**: I is one of {1,2}, J is one of {3,8,11}, K is one of {1,2}, AssLevelComb is one of {low/low, substantial/substantial, high/high}, ConsignmentModeId is one of {basic,consented,consentedSigned}, L is one of {1,2,3} and M is one of {1,2,3}

Would identify all those REM payloads built when:

- RFCFields#I is one of { RFCFields#1, RFCFields#2};
- NewFields#J is one of {NewFields#3, NewFields#8, NewFields#11};
- AssLevelComb is one of the following combinations {low/low, substantial/substantial, high/high};
- ConsignmentModeId is one of {basic,consented,consentedSigned};
- FreeText#K is one of {FreeText#1, FreeText#2};
- OrMess#L is one of {OrMess#1, OrMess#2, OrMess#3}; and
- Sig#M is one of {Sig#1, Sig#2, Sig#3}.

Cells in "Purpose" column contain a description of the purpose of the REM payload instance.

Table 19: Instances of REM payload

Instance	Purpose
REM_payloadInst (OuterMostHeader(RFCFields#I,NewFields#J,nil,nil),REMSIntrCom,FreeText#K,HTMLComb,OrMess#L,Sig#M) **Where**: I is one of {1,2}, J is one of {1,6,9}, K is one of {1,2}, L is one of {1,2,3} and M is one of {1,2,3}	No assurance levels indication. No consignment indication. No extension as there are not structured-valued fields in the headers REM payloads where J is 1: Will be used in scenarios where there is relaying of REM messages, as the REM-RelayDate header field is present. Incorporate REM-RelayDate header field. REM payloads where J is 6: Will be used in scenarios where there is relaying of REM messages, as the REM-RelayDate header field is present. Do not incorporate REM-RelayDate header field. REM payloads where J is 9: Cannot be used in scenarios where there is relaying of REM messages, as the REM-RelayDate header field is absent. Do not incorporate REM-RelayDate header field.

Instance	Purpose
REM_payloadInst (OuterMostHeader(RFCFields#I,NewFields#J,AssLevelComb,nil),REMSIntrCom,FreeText#K,HTMLComb,OrMess#L,Sig#M) **Where**: I is one of {1,2}, J is one of {2,7,10}, K is one of {1,2}, AssLevelComb is one of {low/low, substantial/substantial, high/high}, L is one of {1,2,3} and M is one of {1,2,3}	Assurance levels indications (either low/low, or.substantial/substantial, or high/high) depending of the value of parameter AssLevelComb. No consignment indication. No extensions. REM payloads where J is 2: Will be used in scenarios where there is relaying of REM messages, as the REM-RelayDate header field is present. Incorporate REM-RelayDate header field. REM payloads where J is 7: Will be used in scenarios where there is relaying of REM messages, as the REM-RelayDate header field is present. Do not incorporate REM-RelayDate header field. REM payloads where J is 10: Cannot be used in scenarios where there is relaying of REM messages, as the REM-RelayDate header field is absent. Do not incorporate REM-RelayDate header field.

Instance	Purpose
REM_payloadInst (OuterMostHeader(RFCFields#I,NewFields#J,AssLevelComb,ConsignmentModeId),REMSIntrCom,FreeText#K,HTMLComb,OrMess#L,Sig#M) **Where**: I is one of {1,2}, J is one of {3,8,11}, K is one of {1,2}, AssLevelComb is one of {low/low, substantial/substantial, high/high}, ConsignmentModeId is one of {basic,consented,consentedSigned}, L is one of {1,2,3} and M is one of {1,2,3}	Assurance levels indications (either low/low, or.substantial/substantial, or high/high) depending of the value of parameter AssLevelComb. Consignment mode indication (either basic, or consented, or consentedSigned) depending of the value of parameter ConsignmentModeId. No extensions. REM payloads where J is 3: Will be used in scenarios where there is relaying of REM messages, as the REM-RelayDate header field is present. Incorporate REM-RelayDate header field. REM payloads where J is 8: Will be used in scenarios where there is relaying of REM messages, as the REM-RelayDate header field is present. Do not incorporate REM-RelayDate header field. REM payloads where J is 11: Cannot be used in scenarios where there is relaying of REM messages, as the REM-RelayDate header field is absent. Do not incorporate REM-RelayDate header field.

Instance	Purpose
REM_payloadInst (OuterMostHeader(RFCFields#I,NewFields#4,AssLevelComb,nil),REMSIntrCom,FreeText#K, HTMLComb,OrMess#L,Sig#M) **Where**: I is one of {1,2}, J is one of {4,5}, AssLevelComb is one of {low/low, substantial/substantial, high/high}, L is one of {1,2,3} and M is one of {1,2,3} REM_payloadInst (OuterMostHeader(RFCFields#I,NewFields#5,AssLevelComb,ConsignmentModeId),REMSIntrCom,FreeText#K, HTMLComb,OrMess#L,Sig#M) **Where**: I is one of {1,2}, J is one of {4,5}, AssLevelComb is one of {low/low, substantial/substantial, high/high}, ConsignmentModeId is one of {basic,consented,consentedSigned}, L is one of {1,2,3} and M is one of {1,2,3}	Payloads for negative test cases. All of them incorporate REM-RelayDate header field. Negative test cases because scheduled delivery time is after the expiration date. No consignment indication. No extensions. Will be used in scenarios where there is relaying of REM messages, as the REM-RelayDate header field is present. REM payloads where J is 4: No consignment mode indication. REM payloads where J is 5: Consignment mode indication as per value of parameter ConsignmentModeId.

6.4 Instances of REMS notification

Table 20 defines instances of REM notifications for being used in test cases. Columns are as in clause 6.3.

The notation for the content of cells in column "Instances" is as indicated in clause 6.3, with the addition that the instances of REMS notifications include Extension MIME sections, and the identification of the specific Extension MIME section follows the same principles as other MIME sections.

Table 20: Instances of REMS notification

Instance	Purpose
REMS_NotificationInst (OuterMostHeader(RFCFields#I,NewFields#J,nil,nil),REMSIntrComb,FreeText#K,HTMLComb,Ext#L,Sig#M) **Where**: I is one of {1,2}, J is one of {1,6,9}, K is one of {1,2}, L is one of {1,2,3}, and M is one of {1,2,3}	No assurance levels indication. No consignment indication. REMS notifications where J is 1: Will be used in scenarios where there is relaying of REM messages, as the REM-RelayDate header field is present. Incorporate REM-RelayDate header field. REMS notifications where J is 6: Will be used in scenarios where there is relaying of REM messages, as the REM-RelayDate header field is present. Do not incorporate REM-RelayDate header field. REMS notifications where J is 9: Cannot be used in scenarios where there is relaying of REM messages, as the REM-RelayDate header field is absent. Do not incorporate REM-RelayDate header field.

Instance	Purpose
REMS_NotificationInst (OuterMostHeader(RFCFields#I,NewFields#J,nil,nil),REMSIntrComb,FreeText#K,HTMLComb,Sig#L) **Where**: I is one of {1,2}, J is one of {1,6,9}, K is one of {1,2}, and L is one of {1,2,3}	No assurance levels indication. No consignment indication. No extension as there are not structured-valued fields in the headers REMS notifications where J is 1: Will be used in scenarios where there is relaying of REM messages, as the REM-RelayDate header field is present. Incorporate REM-RelayDate header field. REMS notifications where J is 6: Will be used in scenarios where there is relaying of REM messages, as the REM-RelayDate header field is present. Do not incorporate REM-RelayDate header field. REMS notifications where J is 9: Cannot be used in scenarios where there is relaying of REM messages, as the REM-RelayDate header field is absent. Do not incorporate REM-RelayDate header field.

Instance	Purpose
REMS_notificationInst (OuterMostHeader(RFCFields#I,NewFields#J,AssLevelComb,nil),REMSIntrCom,FreeText#K,HTMLComb,Ext#L,Sig#M) **Where**: I is one of {1,2}, J is one of {2,7,10}, K is one of {1,2}, AssLevelComb is one of {low/low, substantial/substantial, high/high}, L is one of {1,2,3}, and M is one of {1,2,3}	Assurance levels indications (either low/low, or.substantial/substantial, or high/high) depending of the value of parameter AssLevelComb. No consignment indication. REMS notifications where J is 2: Will be used in scenarios where there is relaying of REM messages, as the REM-RelayDate header field is present. Incorporate REM-RelayDate header field. REMS notifications where J is 7: Will be used in scenarios where there is relaying of REM messages, as the REM-RelayDate header field is present. Do not incorporate REM-RelayDate header field. REMS notifications where J is 10: Cannot be used in scenarios where there is relaying of REM messages, as the REM-RelayDate header field is absent. Do not incorporate REM-RelayDate header field.

Instance	Purpose
REMS_notificationInst (OuterMostHeader(RFCFields#I,NewFields#J,AssLevelComb,nil),REMSIntrCom,FreeText#K,HTMLComb, Sig#L) **Where**: I is one of {1,2}, J is one of {2,7,10}, K is one of {1,2}, AssLevelComb is one of {low/low, substantial/substantial, high/high}, and L is one of {1,2,3}	Assurance levels indications (either low/low, or.substantial/substantial, or high/high) depending of the value of parameter AssLevelComb. No consignment indication. No extensions. REMS notifications where J is 2: Will be used in scenarios where there is relaying of REM messages, as the REM-RelayDate header field is present. Incorporate REM-RelayDate header field. REMS notifications where J is 7: Will be used in scenarios where there is relaying of REM messages, as the REM-RelayDate header field is present. Do not incorporate REM-RelayDate header field. REMS notifications where J is 10: Cannot be used in scenarios where there is relaying of REM messages, as the REM-RelayDate header field is absent. Do not incorporate REM-RelayDate header field.

6.5 Instances of REMS receipts

Table 21 defines instances of REMS receipts for being used in test cases. Columns are as in clause 6.3.

A REMS receipt can have more than one ERDS evidence MIME section. The number and contents of these MIME sections will depend on the specific test case.

Table 21: Instances of REMS receipt

Instance	Purpose
REMS_ReceiptInst (OuterMostHeader(RFCFields#I,NewFields#J,nil,nil),REMSIntrComb,FreeText#K,HTMLComb,Ext#L, <EVID#M>+,Sig#N) **Where**: I is one of {1,2}, J is one of {1,6,9}, K is one of {1,2}, L is one of {1,2,3}, M is one of {1,2,3}, and L is one of {1,2,3}. Additionally, the + symbol in <EVID#M> indicates that in each case, the REMS receipt instance shall contain as many Evidence MIME sections as Evidence MIME sections are indicated in the test case where the REMS receipt is used. This unique content identifies all the possible REMS receipts that will be needed in the definitions of the test cases.	No assurance levels indication. No consignment indication. REMS receipts where J is 1: Will be used in scenarios where there is relaying of REM messages, as the REM-RelayDate header field is present. Incorporate REM-RelayDate header field. REMS receipts where J is 6: Will be used in scenarios where there is relaying of REM messages, as the REM-RelayDate header field is present. Do not incorporate REM-RelayDate header field. REMS receipts where J is 9: Cannot be used in scenarios where there is relaying of REM messages, as the REM-RelayDate header field is absent. Do not incorporate REM-RelayDate header field.

6.6 Instances of REM dispatch

Table 22 defines instances of REM dispatches for being used in test cases. Columns are as in clause 6.3. Details of the ERDS evidence MIME sections are indicated as in clause 6.5.

Table 22: Instances of REM dispatch

Instance	Purpose
REM_dispatchInst (OuterMostHeader(RFCFields#I,NewFields#J,nil,nil),REMSIntrCom,FreeText#K,HTMLComb,OrMess#L, Ext#M,<EVID#N>+,Sig#O) **Where**: I is one of {1,2}, J is one of {1,6,9}, K is one of {1,2}, L is one of {1,2,3}, M is one of {1,2,3}, N is one of {1,2,3}, and O is one of {1,2,3}. As before the symbol + in <EVID#M>+ indicates that there will be as many Evidence MIME sections as the sections indicated in the test case using this REM dispatch.	No assurance levels indication. No consignment indication. No extensions REM dispatches where J is 1: Will be used in scenarios where there is relaying of REM messages, as the REM-RelayDate header field is present. Incorporate REM-RelayDate header field. REM dispatches where J is 6: Will be used in scenarios where there is relaying of REM messages, as the REM-RelayDate header field is present. Do not incorporate REM-RelayDate header field. REM dispatches where J is 9: Cannot be used in scenarios where there is relaying of REM messages, as the REM-RelayDate header field is absent. Do not incorporate REM-RelayDate header field.

Instance	Purpose
REM_dispatchInst (OuterMostHeader(RFCFields#I,NewFields#J,AssLevelComb,nil),REMSIntrCom,FreeText#K,HTMLComb,OrMess#L,Ext#M,<EVID#N>+,Sig#O) **Where**: I is one of {1,2}, J is one of {2,7,10}, K is one of {1,2}, AssLevelComb is one of {low/low, substantial/substantial, high/high}, L is one of {1,2,3}, M is one of {1,2,3}, N is one of {1,2,3}, and O is one of {1,2,3}. As before the symbol + in <EVID#M>+ indicates that there will be as many Evidence MIME sections as the sections indicated in the test case using this REM dispatch.	Assurance levels indications (either low/low, or.substantial/substantial, or high/high) depending of the value of parameter AssLevelComb. No consignment indication. No extensions. REM dispatches where J is 2: Will be used in scenarios where there is relaying of REM messages, as the REM-RelayDate header field is present. Incorporate REM-RelayDate header field. REM dispatches where J is 7: Will be used in scenarios where there is relaying of REM messages, as the REM-RelayDate header field is present. Do not incorporate REM-RelayDate header field. REM dispatches where J is 10: Cannot be used in scenarios where there is relaying of REM messages, as the REM-RelayDate header field is absent. Do not incorporate REM-RelayDate header field.

Instance	Purpose
REM_dispatchInst OuterMostHeader(RFCFields#I,NewFields#J,AssLevelComb, ConsignmentModeId),REMSIntrCom,FreeText#K,HTMLComb,OrMess#L,Ext#M,<EVID#N>+,Sig#O) **Where**: I is one of {1,2}, J is one of {3,8,11}, K is one of {1,2}, AssLevelComb is one of {low/low, substantial/substantial, high/high}, ConsignmentModeId is one of {basic,consented,consentedSigned}, L is one of {1,2,3}, M is one of {1,2,3}, N is one of {1,2,3}, and O is one of {1,2,3}. As before the symbol + in <EVID#M>+ indicates that there will be as many Evidence MIME sections as the sections indicated in the test case using this REM dispatch.	Assurance levels indications (either low/low, or.substantial/substantial, or high/high) depending of the value of parameter AssLevelComb. Consignment mode indication (either basic, or consented, or consentedSigned) depending of the value of parameter ConsignmentModeId. No extensions. REM dispatches where J is 3: Will be used in scenarios where there is relaying of REM messages, as the REM-RelayDate header field is present. Incorporate REM-RelayDate header field. REM dispatches where J is 8: Will be used in scenarios where there is relaying of REM messages, as the REM-RelayDate header field is present. Do not incorporate REM-RelayDate header field. REM dispatches where J is 11: Cannot be used in scenarios where there is relaying of REM messages, as the REM-RelayDate header field is absent. Do not incorporate REM-RelayDate header field.

Instance	Purpose
REM_payloadInst (OuterMostHeader(RFCFields#I,NewFields#4,AssLevelComb,nil),REMSIntrCom,FreeText#K, HTMLComb,OrMess#L,Sig#M) **Where**: I is one of {1,2}, J is one of {4,5}, AssLevelComb is one of {low/low, substantial/substantial, high/high}, L is one of {1,2,3} and M is one of {1,2,3} REM_payloadInst (OuterMostHeader(RFCFields#I,NewFields#5,AssLevelComb,ConsignmentModeId),REMSIntrCom,FreeText#K, HTMLComb,OrMess#L,Sig#M) **Where**: I is one of {1,2}, J is one of {4,5}, AssLevelComb is one of {low/low, substantial/substantial, high/high}, ConsignmentModeId is one of {basic,consented,consentedSigned}, L is one of {1,2,3} and M is one of {1,2,3}	Payloads for negative test cases. All of them incorporate REM-RelayDate header field. Negative test cases because scheduled delivery time is after the expiration date. No consignment indication. No extensions. Will be used in scenarios where there is relaying of REM messages, as the REM-RelayDate header field is present. REM payloads where J is 4: No consignment mode indication. REM payloads where J is 5: Consignment mode indication as per value of parameter ConsignmentModeId.

7 Other named sets

7.1 Named sets of participating REMSs

The present clause defines named sets of participating REMSs that are used for naming test cases in clause 5.

The details on the participating REMSs are given using the following pattern: REMSs(I,J), where:

- I stands for the number of Intermediate REMSs (IREMSs); and
- J stands for the number of the Recipient's REMSs (RREMSs).

7.2 Named sets of additional requirements

The present clause defines named sets of additional requirements that are used for building different test cases based on the same scenarios of REM messages.

Table 23 shows the named sets of additional requirements.

Table 23: Named sets of additional requirements

Name of the set	Additional requirements in the set
AdditionalReqs#1	Original message: not signed, no attachment
	Number of recipients: 1
	No sender's delegate
	Sender sends original message
AdditionalReqs#2	Original message: not signed, no attachment
	Number of recipients: 1
	Sender's delegate
	Sender's delegate sends original message

7.3 Named sets of entities in receiving side

The present clause defines named sets of entities that are present at receiving side. This allows using one scenario in defining different test cases by changing the entities in the receiving side.

EXAMPLE: Scenarios defined for one recipient could be used in test cases where the scenarios involve only one delegate of one recipient.

In order to define the names of the sets, the pattern RecSide(I,J,K) is used where:

- I stands for the number of recipients.
- J stands for the number of recipient's delegates.
- K stands for the number of recipients each delegate is delegate of.

K shall always be less or equal than I. If I is not 0 then K shall also be different from 0.

8 Test cases definition

8.1 Introduction

The notations shown in clauses 4.1, 6 and 7, allow building a compact notation for defining tests cases.

The present document defines sets of test cases. Each set of test cases is expressed as a function of a number of parameters (some of them are integers, other are tuples of several values, other -mainly reasons for failures- are enumerated values specified in another ETSI deliverable).

Under these conditions one specific test case is obtained when the set is particularized by assigning a single value to each parameter.

For helping in understanding the notation, below follows the definition of the set of test cases for the scenario REM_SF#3. The definition of a set of test cases has two parts. Below follows the first one:

```
REM_SF#3(
RecSide(F,G,H),
REM_dispatchInst_with_XML_SUB_ACC
(OuterMostHeader(RFCFields#I, NewFields#J, AssLevelComb,
ConsignmentModeId),REMSIntrCom,FreeText#K,HTMLComb,OrMess#L, Ext#M,<EVID#N>,Sig#O),
REMS_receipt_with_XML_SUB_REJ
(OuterMostHeader(RFCFields#I, NewFields#J, AssLevelComb,
ConsignmentModeId),REMSIntrComb,FreeText#K,HTMLComb,Ext#L,<EVID#M>+,Sig#N),
AdditionalReqs#P
)
```

This part shows the components required for defining the test cases for this scenario. For this scenario each test case in the set will require providing details of:

1) The entities in the receiving side (RecSide). The notation for identifying one of the different alternatives is as specified in clause RECEIVING SIDE.

2) The REM Dispatch instance, which also carries an ERDS Evidence (XML_SUB_ACC). The notation for completely defining one specific instance among all the possibilities offered by the different parameters, is as specified in clause REMDISPATCH.

3) The REMS Receipt, carrying a XML_SUB_ACC ERDS evidence. The notation for completely defining one specific instance among all the possibilities offered by the different parameters, is as specified in clause REMSEVIDENCE.

4) The additional requirements, whose notation has been specified in clause ADDITIONALREQS.

Some scenarios include REM payloads instead of REM dispatches. The details of the components of a REM payload are provided as the details of components of a REM dispatch.

Below follows the second part of the definition of the set of test cases:

Where:

- For RecSide: (F,G,H) is one of {(1,0,0), (3,0,0), (3,1,2),(3,1,3)}: define test cases for one recipient, several recipients, and several recipients and one recipient's delegate.

- For REM_dispatchInst_with_XML_SUB_ACC and REMS_receipt_with_{XML_SUB_ACC , XML_CONT_CONS}:

    - I is one of {1,2}, J is one of {9,10,11}, AssLevelComb is one of {nil, low/low, substantial/substantial, high/high}, ConsignmentModeId is one of {nil, basic, consented, consentedSigned}, L is one of {1,2,3}, M is one of {1,2,3}, N is one of {1,2,3}, and O is one of {1,2,3}. For AdditionalReqs: P is one of {1,2}.

This part shows the different values that the parameters present in the first part, can have.

Each legal combination of all the parameters will collapse the set in ONE test case. For instance

```
RecSide(1,0,0),
REM_dispatchInst_with_XML_SUB_ACC
(OuterMostHeader(RFCFields#1, NewFields#11, high/high,
consentedSigned),REMSIntrCom,FreeText#1,HTMLComb,OrMess#1, Ext#1,<EVID#1>,Sig#1),
REMS_receipt_with_XML_SUB_REJ
(OuterMostHeader(RFCFields#1, NewFields#9 nil,
nil),REMSIntrComb,FreeText#1,HTMLComb,Ext#1,<EVID#1>+,Sig#1),
AdditionalReqs#P
```

Defines ONE test case in the set, where:

- The REM-RelayDate and REM-ScheduledDelivery header fields are absent in the outermost header of the REM Dispatch and the REMS Receipt (NewFields#11 combination of new header fields).
- The assurance level combination indication is present and its value is high/high.
- The consignment mode indication is present and its value is consentedSigned.
- The headers in Free Text, Extension, Evidence, and Signature MIME extensions are as indicated in the corresponding clauses defining combinations of header fields for these MIME sections, when the parameter in all of them is 1.

8.2 Test cases for black-box model

8.2.1 Test cases for Store&Forward style of operation

8.2.1.1 Introduction

The present clause defines a set of test cases for each scenario defined in clause XXX (Store&Forward style of operation in the black-box model).

Below follow some remarks that apply to all the sets defined in the present clause:

- All the scenarios do not deal with relaying of REM messages. REM-RelayDate header field is absent (i.e. the valid combination of NewFields are #9 to #11).
- Each set includes test cases for different sets of entities at the receiving side (one or several recipients, and several recipients and one recipients' delegate).
- Each set includes test cases for sender and test cases for sender's delegate.

Below follow a set of rules that govern the values of NewFields#J, Assurance levels indication, and Consignment mode indication:

1) Absence of Assurance levels indication is indicated by a nil value.
2) Absence of Consignment mode indication is indicated by a nil value.
3) For NewFields#9 neither Assurance levels indication nor Consignment mode indication are present.
4) For NewFields#10 Assurance levels indication is present, and Consignment mode indication is absent.
5) For NewFields#11 both Assurance levels indication, and Consignment mode indication are present.

Any combination (NewFields#J, AssLevelComb, ConsignmentModeId) in a specific test case has to meet the rules 3 to 4.

This clause defines, for this model and style of operation, one set of test cases with the details shown above.

For other styles and models, the present document will present rules for defining such sets, as defining one set for each scenario would make the present document extremely long, and the specific definition of one set of test cases for one specific scenario will be straightforward applying the aforementioned rules.

8.2.1.2 Test cases for scenario REM_SF#1

```
REM_SF#1(
RecSide(F,G,H),
REMS_receipt_with_XML_SUB_REJ
(OuterMostHeader(RFCFields#I,NewFields#J, AssLevelComb,
ConsignmentModeId),REMSIntrComb,FreeText#K,HTMLComb,Ext#L,<EVID#M>+,Sig#N),
AdditionalReqs#P,
FailureReason
)
```

Where:

- For RecSide: (F,G,H) is one of {(1,0,0), (3,0,0), (3,1,2),(3,1,3)}: define test cases for one recipient, several recipients, and several recipients and one recipient's delegate.
- For REMS_receipt_with_XML_SUB_REJ:
    - I is one of {1,2}, J is one of {9,10,11}, AssLevelComb is one of {nil, low/low, substantial/substantial, high/high}, ConsignmentModeId is one of {nil, basic, consented, consentedSigned }, L is one of {1,2,3}, M is one of {1,2,3}, and N is one of {1,2,3}.
- For AdditionalReqs: P is one of {1,2}.
- FailureReason is one of {RA02,RA03,RD04,RA05 } as defined in clause 8.3.3.1 of ETSI EN 319 522-2 [2].

8.2.1.3 Test cases for scenario REM_SF#2

```
REMSF#2(
RecSide(F,G,H),
REMS_receipt_with_{2 XML_SUB_REJ}
(OuterMostHeader(RFCFields#I,NewFields#J, AssLevelComb,
ConsignmentModeId),REMSIntrComb,FreeText#K,HTMLComb,Ext#L,<EVID#M>+,Sig#N),
AdditionalReqs#P
FailureReason
)
```

Where:

- For RecSide: (F,G,H) is one of {(1,0,0), (3,0,0), (3,1,2),(3,1,3)}: define test cases for one recipient, several recipients, and several recipients and one recipient's delegate.
- For REMS_receipt_with_{**2** XML_SUB_REJ}:
    - I is one of {1,2}, J is one of {9,10,11}, AssLevelComb is one of {nil, low/low, substantial/substantial, high/high}, ConsignmentModeId is one of {nil, basic, consented, consentedSigned }, K is one of {1,2,3}, L is one of {1,2,3}, M is one of {1,2,3}, and N is one of {1,2,3}.
- For AdditionalReqs: P is one of {1,2}.
- FailureReason is one of {RA02,RA03,RD04,RA05 } as defined in clause 8.3.3.1 ETSI EN 319 522-2 [2].. Failure reasons in REMS receipts will be different.

8.2.1.4 Test cases for scenario REM_SF#3

```
REM_SF#3(
RecSide(F,G,H),
REM_dispatchInst_with_XML_SUB_ACC
(OuterMostHeader(RFCFields#I, NewFields#J, AssLevelComb,
ConsignmentModeId),REMSIntrCom,FreeText#K,HTMLComb,OrMess#L, Ext#M,<EVID#N>,Sig#O),
REMS_receipt_with_XML_SUB_REJ
(OuterMostHeader(RFCFields#I, NewFields#J, AssLevelComb,
ConsignmentModeId),REMSIntrComb,FreeText#K,HTMLComb,Ext#L,<EVID#M>+,Sig#N),
AdditionalReqs#P
)
```

Where:

- For RecSide: (F,G,H) is one of {(1,0,0), (3,0,0), (3,1,2),(3,1,3)}: define test cases for one recipient, several recipients, and several recipients and one recipient's delegate.
- For REM_dispatchInst_with_XML_SUB_ACC and REMS_receipt_with_{XML_SUB_ACC , XML_CONT_CONS}:
  - I is one of {1,2}, J is one of {9,10,11}, AssLevelComb is one of {nil, low/low, substantial/substantial, high/high}, ConsignmentModeId is one of {nil, basic, consented, consentedSigned}, L is one of {1,2,3}, M is one of {1,2,3}, N is one of {1,2,3}, and O is one of {1,2,3}.
- For AdditionalReqs: P is one of {1,2}.

8.2.1.5 Test cases for scenario REM_SF#4

```
REM_SF#4(
RecSide(F,G,H),
REM_dispatchInst_with_XML_SUB_ACC
(OuterMostHeader(RFCFields#I, NewFields#J, AssLevelComb,
ConsignmentModeId),REMSIntrCom,FreeText#K,HTMLComb,OrMess#L, Ext#M,<EVID#N>,Sig#O),
REMS_receipt_with_{XML_SUB_ACC , XML_CONT_CONS, XML_CONT_CONS_FAIL}
(OuterMostHeader(RFCFields#I, NewFields#J, AssLevelComb,
ConsignmentModeId),REMSIntrComb,FreeText#K,HTMLComb,Ext#L,<EVID#M>+,Sig#N),
AdditionalReqs#P,
FailureReason
)
```

Where:

- For RecSide: (F,G,H) is one of {(1,0,0), (3,0,0), (3,1,2),(3,1,3)}: define test cases for one recipient, several recipients, and several recipients and one recipient's delegate.
- For REM_dispatchInst_with_XML_SUB_ACC and REMS_receipt_with_{XML_SUB_ACC , XML_CONT_CONS, XML_CONT_CONS_FAIL}:
  - I is one of {1,2}, J is one of {9,10,11}, AssLevelComb is one of {nil, low/low, substantial/substantial, high/high}, ConsignmentModeId is one of {nil, basic, consented, consentedSigned }, L is one of {1,2,3}, M is one of {1,2,3}, N is one of {1,2,3}, and O is one of {1,2,3}.
- For AdditionalReqs: P is one of {1,2}.
- FailureReason is one of {RD03,RD04,RD05,RD06} as defined in clause 8.3.3.4 of ETSI EN 319 522-2 [2].

8.2.1.6 Test cases for scenario REM_SF#5

```
REM_SF#5(
RecSide(F,G,H),
REM_dispatchInst_with_XML_SUB_ACC
(OuterMostHeader(RFCFields#I, NewFields#J, AssLevelComb,
ConsignmentModeId),REMSIntrCom,FreeText#K,HTMLComb,OrMess#L, Ext#M,<EVID#N>,Sig#O),
REMS_receipt_with_{XML_SUB_ACC , XML_CONT_CONS, 2 XML_CONT_CONS_FAIL}
(OuterMostHeader(RFCFields#I, NewFields#J, AssLevelComb,
ConsignmentModeId),REMSIntrComb,FreeText#K,HTMLComb,Ext#L,<EVID#M>+,Sig#N),
AdditionalReqs#P,
FailureReason
)
```

Where:

- For RecSide: (F,G,H) is one of {(1,0,0), (3,0,0), (3,1,2),(3,1,3)}: define test cases for one recipient, several recipients, and several recipients and one recipient's delegate.
- For REM_dispatchInst_with_XML_SUB_ACC and REMS_receipt_with_{XML_SUB_ACC , XML_CONT_CONS, **2** XML_CONT_CONS_FAIL}:
  - I is one of {1,2}, J is one of {9,10,11}, AssLevelComb is one of {nil, low/low, substantial/substantial, high/high}, ConsignmentModeId is one of {nil, basic, consented, consentedSigned }, L is one of {1,2,3}, M is one of {1,2,3}, N is one of {1,2,3}, and O is one of {1,2,3}.
- For AdditionalReqs: P is one of {1,2}.
- FailureReason is one of {RD03,RD04,RD05,RD06} as defined in clause 8.3.3.4 of ETSI EN 319 522-2 [2].

8.2.1.7 Test cases for scenario REM_SF#6

```
REM_SF#6(
RecSide(F,G,H),
REM_dispatchInst_with_XML_SUB_ACC
(OuterMostHeader(RFCFields#I, NewFields#J, AssLevelComb,
ConsignmentModeId),REMSIntrCom,FreeText#K,HTMLComb,OrMess#L, Ext#M,<EVID#N>,Sig#O),
REMS_receipt_with_{XML_SUB_ACC, XML_CONT_CONS, XML_CONT_HAND}
(OuterMostHeader(RFCFields#I, NewFields#J, AssLevelComb,
ConsignmentModeId),REMSIntrComb,FreeText#K,HTMLComb,Ext#L,<EVID#M>+,Sig#N),
AdditionalReqs#P
)
```

Where:

- For RecSide: (F,G,H) is one of {(1,0,0), (3,0,0), (3,1,2),(3,1,3)}: define test cases for one recipient, several recipients, and several recipients and one recipient's delegate.
- For REM_dispatchInst_with_XML_SUB_ACC and REMS_receipt_with_{XML_SUB_ACC , XML_CONT_CONS}:
  - I is one of {1,2}, J is one of {9,10,11}, AssLevelComb is one of {nil, low/low, substantial/substantial, high/high}, ConsignmentModeId is one of {nil, basic, consented, consentedSigned }, L is one of {1,2,3}, M is one of {1,2,3}, N is one of {1,2,3}, and O is one of {1,2,3}.
- For AdditionalReqs: P is one of {1,2}.

8.2.1.8 Test cases for scenario REM_SF#7

```
REM_SF#7(
RecSide(F,G,H),
REM_dispatchInst_with_XML_SUB_ACC
(OuterMostHeader(RFCFields#I, NewFields#J, AssLevelComb,
ConsignmentModeId),REMSIntrCom,FreeText#K,HTMLComb,OrMess#L, Ext#M,<EVID#N>,Sig#O),
REMS_receipt_with_{XML_SUB_ACC, XML_CONT_CONS, XML_CONT_HAND, XML_CONT_HAND_FAIL}
(OuterMostHeader(RFCFields#I, NewFields#J, AssLevelComb,
ConsignmentModeId),REMSIntrComb,FreeText#K,HTMLComb,Ext#L,<EVID#M>+,Sig#N),
AdditionalReqs#P,
FailureReason
)
```

Where:

- For RecSide: (F,G,H) is one of {(1,0,0), (3,0,0), (3,1,2),(3,1,3)}: define test cases for one recipient, several recipients, and several recipients and one recipient's delegate.

- For REM_dispatchInst_with_XML_SUB_ACC and REMS_receipt_with_{XML_SUB_ACC, XML_CONT_CONS, XML_CONT_HAND, XML_CONT_HAND_FAIL}:
  - I is one of {1,2}, J is one of {9,10,11}, AssLevelComb is one of {nil, low/low, substantial/substantial, high/high}, ConsignmentModeId is one of {nil, basic, consented, consentedSigned },L is one of {1,2,3}, M is one of {1,2,3}, N is one of {1,2,3}, and O is one of {1,2,3}.
- For AdditionalReqs: P is one of {1,2}.
- FailureReason is one of {RE03,RE04 } as defined in clause 8.3.3.5 of ETSI EN 319 522-2 [2].

8.2.1.9 Test cases for scenario REM_SF#8

```
REM_SF#8(
RecSide(F,G,H),
REM_dispatchInst_with_XML_SUB_ACC
(OuterMostHeader(RFCFields#I, NewFields#J, AssLevelComb,
ConsignmentModeId),REMSIntrCom,FreeText#K,HTMLComb,OrMess#L, Ext#M,<EVID#N>,Sig#O),
REMS_receipt_with_{XML_SUB_ACC, XML_CONT_CONS, XML_CONT_HAND, 2 XML_CONT_HAND_FAIL}
(OuterMostHeader(RFCFields#I, NewFields#J, AssLevelComb,
ConsignmentModeId),REMSIntrComb,FreeText#K,HTMLComb,Ext#L,<EVID#M>+,Sig#N),
AdditionalReqs#P,
FailureReason
)
```

Where:

- For RecSide: (F,G,H) is one of {(1,0,0), (3,0,0), (3,1,2),(3,1,3)}: define test cases for one recipient, several recipients, and several recipients and one recipient's delegate.
- For REM_dispatchInst_with_XML_SUB_ACC and REMS_receipt_with_{XML_SUB_ACC, XML_CONT_CONS, XML_CONT_HAND, **2** XML_CONT_HAND_FAIL}:
  - I is one of {1,2}, J is one of {9,10,11}, AssLevelComb is one of {nil, low/low, substantial/substantial, high/high}, ConsignmentModeId is one of {nil, basic, consented, consentedSigned }, L is one of {1,2,3}, M is one of {1,2,3}, N is one of {1,2,3}, and O is one of {1,2,3}.
- For AdditionalReqs: P is one of {1,2}.
- FailureReason is one of {RE03,RE04 } as defined in clause 8.3.3.5 of ETSI EN 319 522-2 [2].

8.2.1.10 Test cases for scenario REM_SF#9

```
REM_SF#9(
RecSide(F,G,H),
REM_dispatchInst_with_XML_SUB_ACC
(OuterMostHeader(RFCFields#I, NewFields#J, AssLevelComb,
ConsignmentModeId),REMSIntrCom,FreeText#K,HTMLComb,OrMess#L, Ext#M,<EVID#N>,Sig#O),
REMS_receipt_with_{XML_SUB_ACC,XML_CONS_NOT, XML_CONT_CONS}
(OuterMostHeader(RFCFields#I, NewFields#J, AssLevelComb,
ConsignmentModeId),REMSIntrComb,FreeText#K,HTMLComb,Ext#L,<EVID#M>+,Sig#N),
REMS_notification_of_Consignment
(OuterMostHeader(RFCFields#I,NewFields#10,
AssLevelComb,nil),REMSIntrComb,FreeText#K,HTMLComb,Ext#L,Sig#M),
AdditionalReqs#P
)
```

Where:

- For RecSide: (F,G,H) is one of {(1,0,0), (3,0,0), (3,1,2),(3,1,3)}: define test cases for one recipient, several recipients, and several recipients and one recipient's delegate.

- For REM_dispatchInst_with_XML_SUB_ACC, REMS_receipt_with__{XML_SUB_ACC,XML_CONS_NOT, XML_CONT_CONS}, and REMS_notification_of_Consignment:
  - I is one of {1,2}, J is one of {9,10,11}, AssLevelComb is one of {nil, low/low, substantial/substantial, high/high}, ConsignmentModeId is one of {nil, basic, consented, consentedSigned },L is one of {1,2,3}, M is one of {1,2,3}, N is one of {1,2,3}, and O is one of {1,2,3}.
- For AdditionalReqs: P is one of {1,2}.

8.2.1.11 Test cases for scenario REM_SF#10

```
REM_SF#10(
RecSide(F,G,H),
REM_dispatchInst_with_XML_SUB_ACC
(OuterMostHeader(RFCFields#I, NewFields#J, AssLevelComb,
ConsignmentModeId),REMSIntrCom,FreeText#K,HTMLComb,OrMess#L, Ext#M,<EVID#N>,Sig#O),
REMS_receipt_with_{XML_SUB_ACC, XML_CONT_CONS, XML_CONS_NOT, XML_CONS_NOT_FAIL }
(OuterMostHeader(RFCFields#I, NewFields#J, AssLevelComb,
ConsignmentModeId),REMSIntrComb,FreeText#K,HTMLComb,Ext#L,<EVID#M>+,Sig#N),
REMS_notification_of_Consignment
(OuterMostHeader(RFCFields#I, NewFields#10,
AssLevelComb,nil),REMSIntrComb,FreeText#K,HTMLComb,Ext#L,Sig#M),
AdditionalReqs#P,
ReasonFailure
)
```

Where:

- For RecSide: (F,G,H) is one of {(1,0,0), (3,0,0), (3,1,2),(3,1,3)}: define test cases for one recipient, several recipients, and several recipients and one recipient's delegate.
- For REM_dispatchInst_with_XML_SUB_ACC, REMS_receipt_with_{XML_SUB_ACC, XML_CONT_CONS, XML_CONS_NOT, XML_CONS_NOT_FAIL}, and REMS_notification_of_Consignment:
  - I is one of {1,2}, J is one of {9,10,11}, AssLevelComb is one of {nil, low/low, substantial/substantial, high/high}, ConsignmentModeId is one of {nil, basic, consented, consentedSigned },L is one of {1,2,3}, M is one of {1,2,3}, N is one of {1,2,3}, and O is one of {1,2,3}.
- For AdditionalReqs: P is one of {1,2}.
- FailureReason is RDXX as defined in clause 8.3.3.4 of ETSI EN 319 522-2 [2].

8.2.1.12 Test cases for scenario REM_SF#11

```
REM_SF#11(
RecSide(F,G,H),
REM_dispatchInst_with_XML_SUB_ACC
(OuterMostHeader(RFCFields#I, NewFields#J, AssLevelComb,
ConsignmentModeId),REMSIntrCom,FreeText#K,HTMLComb,OrMess#L, Ext#M,<EVID#N>,Sig#O),
REMS_receipt_with_{XML_SUB_ACC, XML_CONT_CONS, XML_CONS_NOT, 2 XML_CONS_NOT_FAIL }
(OuterMostHeader(RFCFields#I, NewFields#J, AssLevelComb,
ConsignmentModeId),REMSIntrComb,FreeText#K,HTMLComb,Ext#L,<EVID#M>+,Sig#N),
REMS_notification_of_Consignment
(OuterMostHeader(RFCFields#I, NewFields#10,
AssLevelComb,nil),REMSIntrComb,FreeText#K,HTMLComb,Ext#L,Sig#M),
AdditionalReqs#P,
ReasonFailure
)
```

Where:

- For RecSide: (F,G,H) is one of {(1,0,0), (3,0,0), (3,1,2),(3,1,3)}: define test cases for one recipient, several recipients, and several recipients and one recipient's delegate.

- For REM_dispatchInst_with_XML_SUB_ACC, REMS_receipt_with_{XML_SUB_ACC, XML_CONT_CONS, XML_CONS_NOT, 2 XML_CONS_NOT_FAIL }, and REMS_notification_of_Consignment:
    - I is one of {1,2}, J is one of {9,10,11}, AssLevelComb is one of {nil, low/low, substantial/substantial, high/high}, ConsignmentModeId is one of {nil, basic, consented, consentedSigned },L is one of {1,2,3}, M is one of {1,2,3}, N is one of {1,2,3}, and O is one of {1,2,3}.
- For AdditionalReqs: P is one of {1,2}.
- FailureReason is RDXX as defined in clause 8.3.3.4 of ETSI EN 319 522-2 [2].

8.2.1.13 Test cases for scenario REM_SF#12

```
REM_SF#12(
RecSide(F,G,H),
REM_payloadInst
(OuterMostHeader(RFCFields#I,NewFields#J,AssLevelComb,ConsignmentModeId),REMSIntrCom,FreeText#K,HTML
Comb,OrMess#L,Sig#M)
REMS_receipt_with_XML_SUB_ACC,
(OuterMostHeader(RFCFields#I, NewFields#J, AssLevelComb,
ConsignmentModeId),REMSIntrComb,FreeText#K,HTMLComb,Ext#L,<EVID#M>+,Sig#N),
REMS_receipt_with_{XML_SUB_ACC, XML_CONT_CONS}
(OuterMostHeader(RFCFields#I, NewFields#J, AssLevelComb,
ConsignmentModeId),REMSIntrComb,FreeText#K,HTMLComb,Ext#L,<EVID#M>+,Sig#N),
AdditionalReqs#P
)
```

Where:

- For RecSide: (F,G,H) is one of {(1,0,0), (3,0,0), (3,1,2),(3,1,3)}: define test cases for one recipient, several recipients, and several recipients and one recipient's delegate.
- For REM_payloadInst, REMS_receipt_with_XML_SUB_ACC,and REMS_receipt_with_{XML_SUB_ACC, XML_CONT_CONS}:
    - I is one of {1,2}, J is one of {9,10,11}, AssLevelComb is one of {nil, low/low, substantial/substantial, high/high}, ConsignmentModeId is one of {nil, basic, consented, consentedSigned }, K is one of {1,2}, L is one of {1,2,3}, M is one of {1,2,3}, N is one of {1,2,3}, and O is one of {1,2,3}.
- For AdditionalReqs: P is one of {1,2}.

8.2.1.14 Test cases for scenario REM_SF#13

```
REM_SF#13(
RecSide(F,G,H),
REM_payloadInst
(OuterMostHeader(RFCFields#I,NewFields#J,AssLevelComb,ConsignmentModeId),REMSIntrCom,FreeText#K,HTML
Comb,OrMess#L,Sig#M)
REMS_receipt_with_XML_SUB_ACC,
(OuterMostHeader(RFCFields#I, NewFields#J, AssLevelComb,
ConsignmentModeId),REMSIntrComb,FreeText#K,HTMLComb,Ext#L,<EVID#M>+,Sig#N),
REMS_receipt_with_ XML_CONT_CONS
(OuterMostHeader(RFCFields#I, NewFields#J, AssLevelComb,
ConsignmentModeId),REMSIntrComb,FreeText#K,HTMLComb,Ext#L,<EVID#M>+,Sig#N),
AdditionalReqs#P
)
```

Where:

- For RecSide: (F,G,H) is one of {(1,0,0), (3,0,0), (3,1,2),(3,1,3)}: define test cases for one recipient, several recipients, and several recipients and one recipient's delegate.
- For REM_payloadInst, REMS_receipt_with_XML_SUB_ACC,and REMS_receipt_with_{XML_SUB_ACC, XML_CONT_CONS}:
    - I is one of {1,2}, J is one of {9,10,11}, AssLevelComb is one of {nil, low/low, substantial/substantial, high/high}, ConsignmentModeId is one of {nil, basic, consented, consentedSigned }, K is one of {1,2}, L is one of {1,2,3}, M is one of {1,2,3}, N is one of {1,2,3}, and O is one of {1,2,3}.

- For AdditionalReqs: P is one of {1,2}.

8.2.2 Test cases for Store&Notify style of operation

8.2.2.1 Rules for REM messages

The following rules will apply for parametrizing the REM messages, which are components of the sets of test cases for scenarios REM_SN#1 to REM_SN#4:

```
REM dispatches with ERDS Evidences will be built on the following components:
(OuterMostHeader(RFCFields#I, NewFields#J, AssLevelComb,
ConsignmentModeId),REMSIntrCom,FreeText#K,HTMLComb,OrMess#L, Ext#M,<EVID#N>,Sig#O)
```

Where:

- I is one of {1,2}, J is one of {9,10,11}, AssLevelComb is one of {nil, low/low, substantial/substantial, high/high}, ConsignmentModeId is one of {nil, basic, consented, consentedSigned },L is one of {1,2,3}, M is one of {1,2,3}, N is one of {1,2,3}, and O is one of {1,2,3}.

```
REM receipts with ERDS Evidences or sets of ERDS Evidences will be built on the following
components:
REMS_receipt_with_{XML_SUB_ACC, XML_CONT_CONS}
(OuterMostHeader(RFCFields#I, NewFields#J, AssLevelComb,
ConsignmentModeId),REMSIntrComb,FreeText#K,HTMLComb,Ext#L,<EVID#M>+,Sig#N)
```

Where:

- I is one of {1,2}, J is one of {9,10,11}, AssLevelComb is one of {nil, low/low, substantial/substantial, high/high}, ConsignmentModeId is one of {nil, basic, consented, consentedSigned },L is one of {1,2,3}, M is one of {1,2,3}, N is one of {1,2,3}, and O is one of {1,2,3}.

```
REMS_notification_for_Acceptance will be built on the following components:
(OuterMostHeader(RFCFields#I, NewFields#10,
AssLevelComb,nil),REMSIntrComb,FreeText#K,HTMLComb,Ext#L,Sig#M)
```

Where:

- I is one of {1,2}, AssLevelComb is one of {nil, low/low, substantial/substantial, high/high}, L is one of {1,2,3}, and M is one of {1,2,3}.

8.2.2.2 Rules for failure reasons

The following FailureReason codes will be used for the (black-box model/Store&Notify style):

1) In scenario REMS_SN#2, one of the receiving entities rejects consignment. One XML_CONS_REJ ERDS evidence is generated and encapsulated in a REMS receipt. For this scenario the reason for failure code is RC08 as specified in clause 8.3.3.3 of ETSI EN 319 522-2 [2].

2) In scenario REMS_SN#3, one of the receiving entities does not answer in time to the notification for acceptance. One XML_ACC_REJ_EXP ERDS evidence is generated and encapsulated in a REMS receipt. For this scenario the reason for failure code is RC09 as specified in clause 8.3.3.3 of ETSI EN 319 522-2 [2].

8.3 Test cases for 4-corner model

8.3.1 Introduction and general rules

In this model, SREMS relays REM messages to the RREMS.

Below follow a set of rules that govern the values of NewFields#J, Assurance levels indication, and Consignment mode indication:

1) For NewFields#1 and NewFields#4 neither Assurance levels indication nor Consignment mode indication are present.

2) For NewFields#2 and NewFields#5 Assurance levels indication is present, and Consignment mode indication is absent.

3) For NewFields#3 and NewFields#6 both Assurance levels indication, and Consignment mode indication are present.

Any combination (NewFields#J, AssLevelComb, ConsignmentModeId) in a specific test case has to meet the rules 3 to 4.

8.3.2 Test cases for Store&Forward to Store&Forward scenarios

8.3.2.1 Rules for REM messages

The following rules will apply for parametrizing the REM messages, which are components of the sets of test cases for scenarios SREMS_SF_RREMS_SF#1 to SREMS_SF_RREMS_SF#5:

```
REM dispatches that are relayed from SREMS to RREMS and REM dispatches that are generated by RREMS
from REM dispatches that have been relayed by the SREMS will be built on the following components:
(OuterMostHeader(RFCFields#I, NewFields#J, AssLevelComb,
ConsignmentModeId),REMSIntrCom,FreeText#K,HTMLComb,OrMess#L, Ext#M,<EVID#N>,Sig#O)
```

Where:

- I is one of {1,2}, J is one of {1,2,3,4,5,6}, AssLevelComb is one of {nil, low/low, substantial/substantial, high/high}, ConsignmentModeId is one of {nil, basic, consented, consentedSigned },L is one of {1,2,3}, M is one of {1,2,3}, N is one of {1,2,3}, and O is one of {1,2,3}.

```
REM receipts with ERDS Evidences or sets of ERDS Evidences will be built on the following
components:
REMS_receipt_with_{XML_SUB_ACC, XML_CONT_CONS}
(OuterMostHeader(RFCFields#I, NewFields#J, AssLevelComb,
ConsignmentModeId),REMSIntrComb,FreeText#K,HTMLComb,Ext#L,<EVID#M>+,Sig#N)
```

Where:

- I is one of {1,2}, J is one of {1,2,3,4,5,6}, AssLevelComb is one of {nil, low/low, substantial/substantial, high/high}, ConsignmentModeId is one of {nil, basic, consented, consentedSigned },L is one of {1,2,3}, M is one of {1,2,3}, N is one of {1,2,3}, and O is one of {1,2,3}.

```
REMS_notification_for_XY will be built on the following components:
(OuterMostHeader(RFCFields#I, NewFields#10,
AssLevelComb,nil),REMSIntrComb,FreeText#K,HTMLComb,Ext#L,Sig#M)
```

Where:

- I is one of {1,2}, AssLevelComb is one of {nil, low/low, substantial/substantial, high/high}, L is one of {1,2,3}, and M is one of {1,2,3}.

8.3.2.2 Rules for failure reasons

The following FailureReason codes will be used for the (4-corner model/Store&Forward to Store&Forward):

1) In scenario SREMS_SF_RREMS_SF#1, RREMS rejects relaying of the REM dispatch generated by SREMS. One XML_REL_REJ ERDS evidence is generated and encapsulated in a REMS receipt. For this scenario the reason for failure code is one of {RB02, RB03, RB04, RB05, RB06, RB02} as specified in clause 8.3.3.2 of ETSI EN 319 522-2 [2].

2) In scenario SREMS_SF_RREMS_SF#4, one of the handovers by a receiving entity fails. One XML_CONT_HAND_FAIL evidence is generated. For this scenario the reason for failure code is one of {RE03,RE04} as specified in clause 8.3.3.5 of ETSI EN 319 522-2 [2].

3) In scenario SREMS_SF_RREMS_SF#5, one of the consignments to a receiving entity fails. One XML_CONT_CONS_FAIL evidence is generated. For this scenario the reason for failure code is one of {RD04, RD05, RD06} as specified in clause 8.3.3.4 of ETSI EN 319 522-2 [2].

8.3.3 Test cases for Store&Forward to Store&Notify scenarios

8.3.3.1 Rules for REM messages

The rules that will apply for parametrizing the REM messages, which are components of the sets of test cases for scenarios SREMS_SF_RREMS_SN#1 to SREMS_SF_RREMS_SN#6 shall be the same as the rules defined in clause 8.3.2.1 for Store&Forward to Store&Forward scenarios.

8.3.3.2 Rules for failure reasons

The following FailureReason codes will be used for the (4-corner model/Store&Forward to Store&Notify):

1) In scenario SREMS_SF_RREMS_SN#2, one receiving entity rejects consignment. One XML_CONS_REJ ERDS evidence is generated and encapsulated in a REMS receipt. For this scenario the reason for failure code is RC08 as specified in clause 8.3.3.3 of ETSI EN 319 522-2 [2].
2) In scenario SREMS_SF_RREMS_SN#3, one receiving entity does not react in time to the notification for acceptance of consignment. One XML_ACC_REJ_EXP evidence is generated and encapsulated in a REMS receipt. For this scenario the reason for failure code is RC09 as specified in clause 8.3.3.3 of ETSI EN 319 522-2 [2].
3) In scenario SREMS_SF_RREMS_SN#4, one of the consignments to a receiving entity fails. One XML_CONT_CONS_FAIL evidence is generated. For this scenario the reason for failure code is one of {RD04, RD05, RD06} as specified in clause 8.3.3.4 of ETSI EN 319 522-2 [2].
4) In scenario SREMS_SF_RREMS_SN#5, one of the handovers by a receiving entity fails. One XML_CONT_HAND_FAIL ERDS Evidence is generated. For this scenario the reason for failure code is one of {RE03,RE04} as specified in clause 8.3.3.5 of ETSI EN 319 522-2 [2].

8.3.4 Test cases for Store&Notify to Store&Forward scenarios

8.3.4.1 Rules for REM messages

The rules that will apply for parametrizing the REM messages, which are components of the sets of test cases for scenarios SREMS_SN_RREMS_SF#1 to SREMS_SN_RREMS_SF#7 shall be the same as the rules defined in clause 8.3.2.1 for Store&Forward to Store&Forward scenarios.

8.3.4.2 Rules for failure reasons

The following FailureReason codes will be used for the (4-corner model/Store&Notify to Store&Forward):

1) In scenario SREMS_SN_RREMS_SF#2, one receiving entity rejects consignment. One XML_CONS_REJ ERDS evidence is generated and encapsulated in a REMS receipt. For this scenario the reason for failure code is RC08 as specified in clause 8.3.3.3 of ETSI EN 319 522-2 [2].
2) In scenario SREMS_SN_RREMS_SF#4, one of the consignments to a receiving entity fails. One XML_CONT_CONS_FAIL ERDS Evidence is generated. For this scenario the reason for failure code is one of {RD04, RD05, RD06} as specified in clause 8.3.3.4 of ETSI EN 319 522-2 [2].
3) In scenario SREMS_SN_RREMS_SF#5, one of the handovers by a receiving entity fails. One XML_CONT_HAND_FAIL ERDS Evidence is generated. For this scenario the reason for failure code is one of {RE03,RE04} as specified in clause 8.3.3.5 of ETSI EN 319 522-2 [2].

8.4 Test cases for extended model

8.4.1 Test cases for scenarios S&F->S&F->S&F

8.4.1.1 Rules for REM messages

The rules that will apply for parametrizing the REM messages, which are components of the sets of test cases for scenarios SREMS_SF_IREMS_SF_RREMS_SF#1 to SREMS_SF_IREMS_SF_RREMS_SF#5 shall be the same as the rules defined in clause 8.3.2.1 for Store&Forward to Store&Forward scenarios.

8.4.1.2 Rules for failure reasons

The following FailureReason codes will be used for the (extended model/S&F->S&F-S&F):

1) In scenario SREMS_SF_IREMS_SF_RREMS_SF#2, one of the consignments to a receiving entity fails. One XML_CONT_CONS_FAIL ERDS Evidence is generated. For this scenario the reason for failure code is one of {RD04, RD05, RD06} as specified in clause 8.3.3.4 of ETSI EN 319 522-2 [2].

2) In scenario SREMS_SF_IREMS_SF_RREMS_SF#3, one of the handovers by a receiving entity fails. One XML_CONT_HAND_FAIL ERDS Evidence is generated. For this scenario the reason for failure code is one of {RE03,RE04} as specified in clause 8.3.3.5 of ETSI EN 319 522-2 [2].

3) In scenario SREMS_SF_IREMS_SF_RREMS_SF#5, RREMS rejects relaying the REM dispatch received from IREMS. One XML_REL_REJ ERDS Evidence is generated. For this scenario the reason for failure code is one of {RB02, RB03, RB04, RB05, RB06} as specified in clause 8.3.3.2 of ETSI EN 319 522-2 [2].

8.4.2 Test cases for scenarios S&F->S&N->S&F

8.4.2.1 Rules for REM messages

The rules that will apply for parametrizing the REM messages, which are components of the sets of test cases for scenarios SREMS_SF_IREMS_SN_RREMS_SF#1 to SREMS_SF_IREMS_SN_RREMS_SF#3 shall be the same as the rules defined in clause 8.3.2.1 for Store&Forward to Store&Forward scenarios.

8.4.2.2 Rules for failure reasons

The following FailureReason codes will be used for the (extended model/S&F->S&N-S&F):

1) In scenario SREMS_SF_IREMS_SN_RREMS_SF#2, one of the receiving entities rejects consignment. One XML_CONS_REJ ERDS Evidence is generated. For this scenario the reason for failure code is RC08 as specified in clause 8.3.3.3 of ETSI EN 319 522-2 [2].

2) In scenario SREMS_SF_IREMS_SF_RREMS_SF#3, one of the receiving entities rejects consignment, and the handovers by another entity fails. One XML_CONS_REJ and XML_CONT_HAND_FAIL ERDS Evidences are generated. For the XML_CONT_HAND_FAIL ERDS Evidence the reason for failure code is one of {RE03,RE04} as specified in clause 8.3.3.5 of ETSI EN 319 522-2 [2]. For the XML_CONS_REJ ERDS Evidence, the reason for failure code is RC08 as specified in clause 8.3.3.3 of ETSI EN 319 522-2 [2].

History

Document history		
V1.1.1	February 2019	Publication