PD CEN/TS 16326:2013

BSI Standards Publication

Postal Services — Hybrid Mail — Functional Specification for postal registered electronic mail

raising standards worldwide™

National foreword

This Published Document is the UK implementation of CEN/TS 16326:2013.

The UK participation in its preparation was entrusted to Technical Committee SVS/4, Postal services.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

© The British Standards Institution 2013

Published by BSI Standards Limited 2013

ISBN 978 0 580 76199 7

ICS 03.240

Compliance with a British Standard cannot confer immunity from legal obligations.

This Published Document was published under the authority of the Standards Policy and Strategy Committee on 31 January 2013.

Amendments issued since publication

Amd. No.	Date	Text affected

TECHNICAL SPECIFICATION

SPÉCIFICATION TECHNIQUE

TECHNISCHE SPEZIFIKATION

CEN/TS 16326

January 2013

ICS 03.240

English Version

Postal Services - Hybrid Mail - Functional Specification for postal registered electronic mail

Services postaux - Courrier hybride - Spécifications fonctionnelles pour le courrier recommandé électronique

Postalische Dienstleistungen - Hybride Sendungen - Funktionale Spezifikation für elektronische Posteinschreibsendungen

EUROPEAN COMMITTEE FOR STANDARDIZATION
COMITÉ EUROPÉEN DE NORMALISATION
EUROPÄISCHES KOMITEE FÜR NORMUNG

Management Centre: Avenue Marnix 17, B-1000 Brussels

Contents

Page

Foreword

This document (CEN/TS 16326:2013) has been prepared by Technical Committee CEN/TC 331 “Postal services”, the secretariat of which is held by NEN.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. CEN [and/or CENELEC] shall not be held responsible for identifying any or all such patent rights.

Keeping pace with the changing communications market, postal operators are increasingly using new communication and information technologies to move beyond what is traditionally regarded as their core postal business. They are meeting higher customer expectations with an expanded range of products and value-added services.

Standards are important prerequisites for effective postal operations and for interconnecting the global network. The European Committee for Standardization, their Technical Committee 331 "Postal Services" develops and maintains a growing number of standards to improve the exchange of postal-related information between postal operators, postal handling organisations, customers, suppliers and other partners, including various international organisations.

This functional specification has been developed in close relationship with the following technical standards:

— CEN/TS 15121-1:2011;

— ETSI TS 102 640-1.

The use of this Technical Specification as a basis for any implementation is at the risk of the user. Any party intending such use is strongly advised to seek close contact with the appropriate working group, so that it can be kept informed of ongoing work.

The CEN/TS 16326 was originally published as a UPU standard S52-1and was adopted by CEN under the current Memorandum of Understanding between UPU and CEN.

According to the CEN/CENELEC Internal Regulations, the national standards organisations of the following countries are bound to announce this Technical Specification: Austria, Belgium, Bulgaria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Former Yugoslav Republic of Macedonia, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, Netherlands, Norway, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, Switzerland, Turkey and the United Kingdom.

Introduction

This document describes the functional specification for a postal operator to build or implement the postal registered electronic mail (abbreviated hereinafter to PReM) system which can be offered to the customers of the postal operator as part of the Secure electronic Postal Services (SePS).

PReM is an electronic version of the traditional postal registered mail service. It uses state-of-the-art cryptographic technologies to provide strong role authentication, protection of message confidentiality and integrity and to add non-repudiation attributes to evidence of events and operations. Therefore, the goal of a PReM is to enhance the traditional e-mail service so as to provide an end-to-end trusted electronic communication service, encompassing both evidence of submission and delivery between authenticated parties.

The service is embodied in the secured and trusted exchange of electronic mail, as every step of the process is logged for future evidence tracing and any entity involved is authenticated. The PReM service comprises the following features:

— secured message forwarding and delivery: ensures the PReM Message confidentiality (encryption) and integrity (no modification); and authenticity and non-repudiation of users (mailer and addressee) and postal operators (origin and destination). In addition, PReM messages will be securely transported from mailer to addressee/mailee;

— evidence generation: all significant events within a complete operation cycle are traceable. Types of evidence and evidence formats are described in 7.2;

— event notification: notification that a particular event/operation has occurred will be generated and sent to corresponding parties;

— archival of evidence: storage of generated evidence for future attestation.

The implementation of part or all of this functional specification might involve the use of intellectual property which is the subject of patent and/or trademark rights. Since the specification was developed in close relationship with ETSI TS 102 640, these might include rights held by ETSI. It is the responsibility of users of the standard to conduct any necessary searches and to ensure that any pertinent rights are in the public domain, are licensed, or are avoided. CEN/TC 331 cannot accept any responsibility in case of infringement, on the part of users of this document, of any third party intellectual property rights. Nevertheless, document users and owners of such rights are encouraged to advise the Secretariat of the UPU Standards Board or the Secretariat of CEN/TC 331 or CEN/TC 331 WG2 of any explicit claim that any technique or solution described herein is protected by such right in any UPU member country. Any such claims will, without prejudice, be documented in the next update of this standard or otherwise at the discretion of the Standards Board or Secretariat of the CEN/TC 331 WG2.

Annex C of this document lists the intellectual property rights brought to the attention of the UPU Standards Board or CEN/TC 331 WG2 prior to approval of the publication of this version of the standard.

1 Scope

This Technical Specification constitutes the functional specification of a secure electronic postal service, referred to as the postal registered electronic mail or PReM service. PReM provides a trusted and certified electronic mail exchange between mailer, postal operators and addressee/mailee. In addition, evidence of corresponding events and operations within the scope of PReM will be generated and archived for future attestation.

The PReM service is defined by reference to the concepts, schemas and operations defined in CEN/TS 15121-1:2011. It utilises six SePS operational verbs (CheckIntegrity, LogEvent, Postmark, RetrieveResults, Sign and Verify) and the five additional server-side operational verbs (SendMessagetoDestination, Subscribe Notification, UnscbscribeNotification, RejectMessage and ReceiveNotification) to fulfil the operational requirements of a PReM System.

Return of Investment (ROI), market potential, revenues model, business plan and pricing policy are outside the scope of this functional specification. Postal operators are advised to make the necessary marketing study and research prior to considering leasing, procuring or developing such a PReM system in accordance with this functional specification.

2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

a) European Telecommunications Standards Institute (ETSI) Standards

NOTE ETSI standards are available at www.etsi.org.

ETSI TS 102 640-1, *Electronic Signatures and Infrastructures (ESI);Registered Electronic Mail (REM); Part 1: Architecture*

ETSI TS 101 862 V1.3.3 (2006-01), *Qualified Certificate Profile*

b) European Committee for Standardization (CEN)

NOTE CEN standards can be obtained from national standardization institutes of CEN National Members (see http://www.cen.eu)

CEN/TS 15121-1:2011, *Postal Services — Hybrid Mail — Part 1: Secured electronic postal services (SePS) interface specification — Concepts, schemas and operations*

CWA 14169, *Secure Signature-Creation Devices "EAL 4+"*

c) Internet Engineering Task Force Public Key Infrastructure X.509 working group (IETF PKIX)

RFC 5280, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, May 2008, D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R.Housley, W.Polk, available from: http://www.rfc-editor.org/rfc/pdfrfc/rfc5280.txt.pdf

d) Universal Postal Union (UPU) Standards

NOTE UPU standards are available on subscription from the UPU International Bureau: Weltpoststrasse 4, Case postale, 3000 Berne 15, SWITZERLAND; Tel: +41 31 350 3111; Fax: +41 31 350 3110; http://www.upu.int

UPU Technical Standard S43a, *Secured electronic postal services (SePS) interface specification — Part A: Concepts, schemas and operations*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in the UPU Standards glossary, in the documents referred to in Clause 2 and in Bibliography, and the following apply. The definition of other frequently used or particularly important terms as well as other terms introduced in this document is given below.

3.1
Advanced Electronic Signature
signature, uniquely linked to and capable of identifying the signatory, which was created by a signature creation device in the sole control of the signatory and which is linked to data in such a way that subsequent change to such data is detectable

3.2
postal operator
entity officially designated by a UPU member country to operate postal services and to fulfil some or all of the related obligations arising out of the Acts of the UPU in its territory

3.3
postal operator Trust List
list of registered and authenticated postal operators which is maintained by the UPU International Bureau

3.4
Email Client Software
software which supports the creation, sending, reception and storage of messages intended to be or actually transmitted over electronic communication systems in accordance with the Simple Mail Transfer Protocol

3.5
notification
message that informs the involved parties that a PReM operation has been performed or a PReM event has taken place

3.6
PReM Dispatch
PReM Message together with all previously collected evidence related thereto the PReM Message

3.7
PReM Object
electronic message or file(s) that mailer intends to send to addressee/mailee

3.8
PReM Policy Domain
collection of PReM enabled postal operators which belong to a group that it is managed according to agreed rules and regulations agreed by the group

3.9
PReM Message
S/MIME object consisting of one or more PReM Objects

3.10
PReM End User
mailer or addressee/mailee of a PReM System

3.11
Qualified Electronic Signature
Advanced Electronic Signature (AES) which is based on a Qualified Certificate (QC) and which is created by a Secure Signature Creation Device (SSCD)

3.12
web browser
software which enables a user to display and interact with text, images, videos, music and other information typically located on a Web page at a web site on the World Wide Web or a local area network

4 Symbols and abbreviations

For the purposes of this document, the symbols, abbreviations and acronyms given in the UPU Standards glossary, and CEN/TS 15121-1:2011 and the following apply:

HTTPS	Hypertext Transfer Protocol over Secure Socket Layer
PReM	Postal Registered electronic Mail
S/MIME	Secure/Multipurpose Internet Mail Extensions
SMTP	Simple Mail Transfer Protocol
POP3	Post Office Protocol version 3
IMAP4	Internet Message Access Protocol
SOAP	Simple Object Access Protocol
SOA	Service-Oriented Architecture
EDI	Electronic Data Interchange
REM	Registered Electronic Mail
ETSI	European Telecommunications Standards Institute
QES	Qualified Electronic Signature
QC	Qualified Certificate
SSCD	Secure Signature Creation Device

5 Coordinate system

5.1 Conceptual models

The conceptual model of a PReM System comprises of the mailer, addressee/mailee, authorised party, postal operator of origin, postal operator of destination and postal operator trust list distribution point.

Records of mailers and/or addressees/mailees held by postal operators should be stored in a directory server so that registered PReM users can search for other PReM users using criteria such as their email address or common name. Lightweight Directory Access Protocol (LDAP) is recommended as the directory service protocol for building such a repository.

Furthermore, in order to identify a legitimate postal operator of destination, a postal operator Trust List should be maintained and published by the UPU acting as the postal operator Trust List Distribution Point under the rules and regulations of the PReM Policy Domain.

Conditions and policies for postal operators to join the PReM Policy Domain are outside the scope of this technical functional specification.

The mailer and addressee/mailee may subscribe to the PReM services offered by the same or different postal operators. Figure 1 represents the case in which the mailer and addressee/mailee subscribe to the service of the same postal operator, while Figure 2 represents the case in which the mailer and addressee/mailee subscribe to the services of different postal operators.

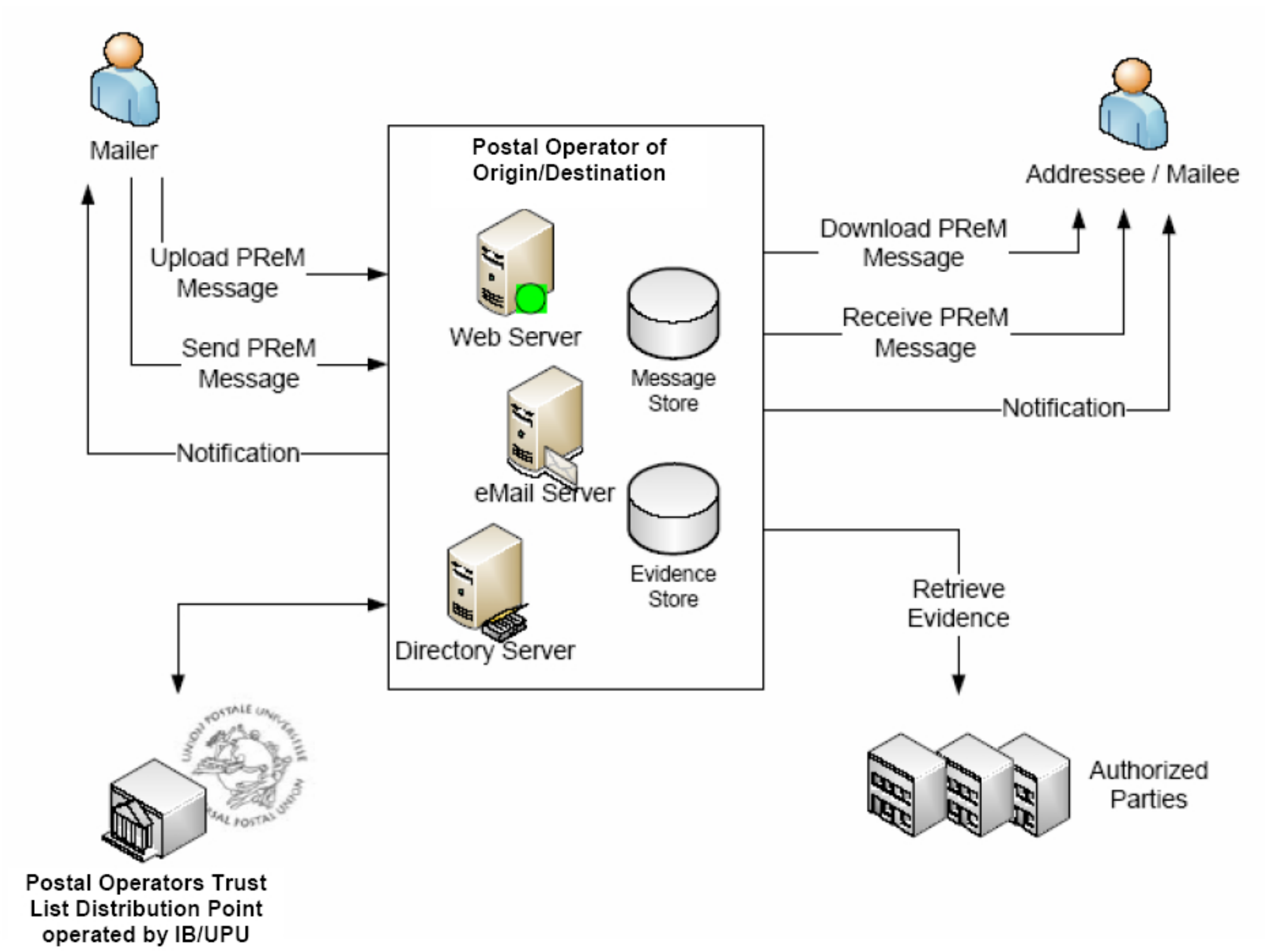

Figure 1 — Mailer and addressee/mailee subscribing to the same postal operator

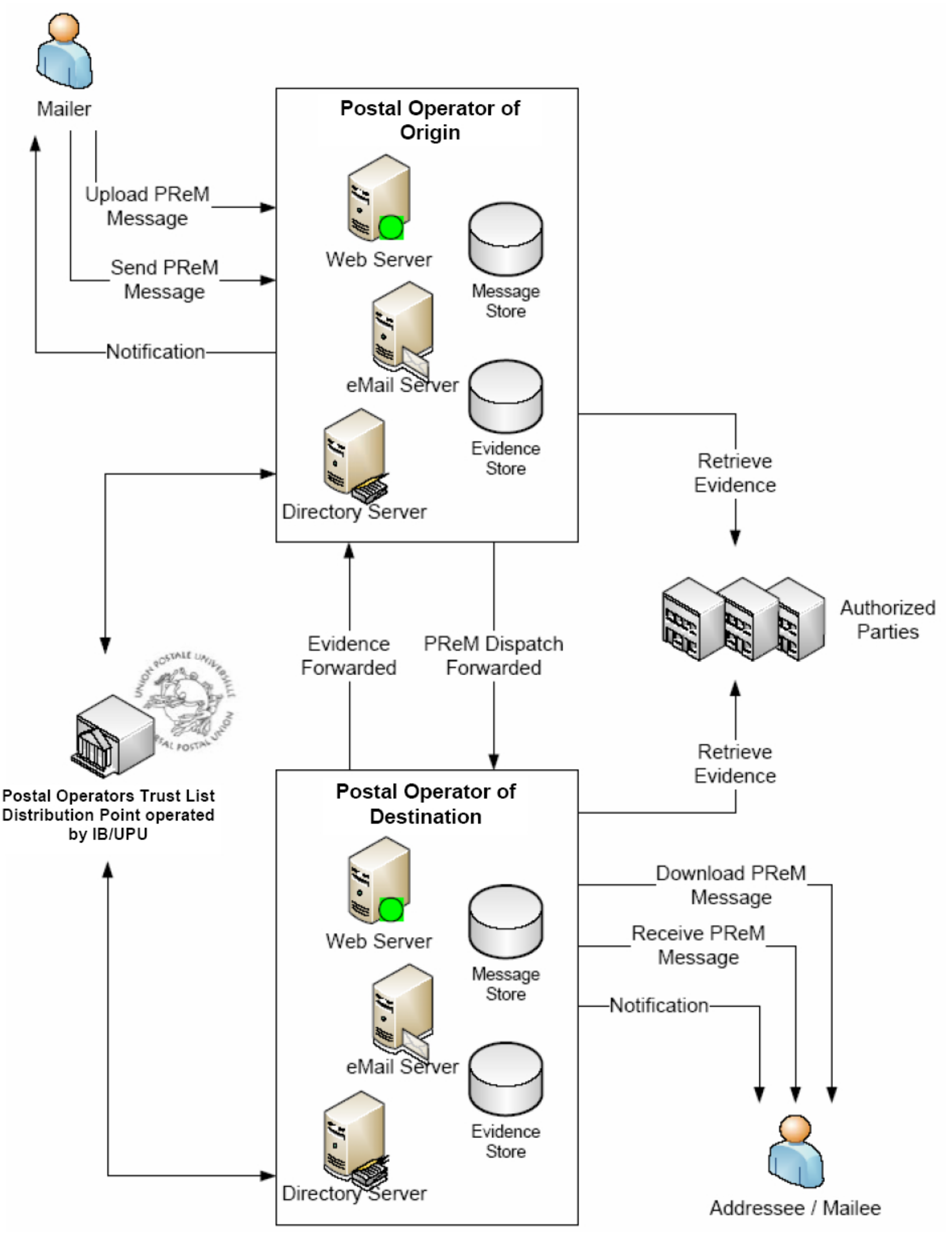

Figure 2 — Mailer and addressee/mailee subscribing to different postal operators

When sending or receiving a PReM Message, mailer and addressee/mailee may use an email client interface or a Web-based Interface to interact with a PReM service provided by a postal operator.

The following table shows all possible interface combinations used by the mailer and addressee/mailee.

Table 1 — possible interface combinations used by the mailer and addressee/mailee mailer

Mailer	Addressee/mailee
Email Client Interface	Email Client Interface
Email Client Interface	Web-based Interface
Web-based Interface	Web-based Interface
Web-based Interface	Email Client Interface

System workflows described in 5.2 only apply when mailer and addressee/mailee subscribe to PReM services of different postal operators. However, if mailer and addressee/mailee subscribe to the PReM service provided by the same postal operator, similar procedures should be applicable with appropriate modifications.

Among all possible interface combinations mentioned in the above table, only the first and third cases are explicitly described in 5.2. However, if the mailer uses Email Client Interface and the addressee/mailee uses Webbased Interface or the mailer uses Web-based Interface and the addressee/mailee uses Email Client Interface, appropriate modifications should be done to the workflow to accommodate these scenarios accordingly.

Although the mailer, addressee/mailee and postal operators are automatically notified with relevant evidence during the transmission process they are entitled to access it at a later date if necessary.

If a mailer so wishes, a PReM Message could be addressed to multiple addressees/mailees, possibly subscribing to a PReM service provided by the same postal operator.

5.2 Operation scenarios

5.2.1 Mailer and addressee/mailee both using an Email Client Interface and subscribing to PReM Service provided by different postal operators

PReM Email Client Interfaces can, but are not required to be, based on a commercially available product such as Microsoft Outlook, Microsoft Outlook Express, Mozilla ThunderBird, Lotus Notes and Qualcomm Eudora. Such products will generally require the user to install Email Client Plug-in software to add PReM functionality.

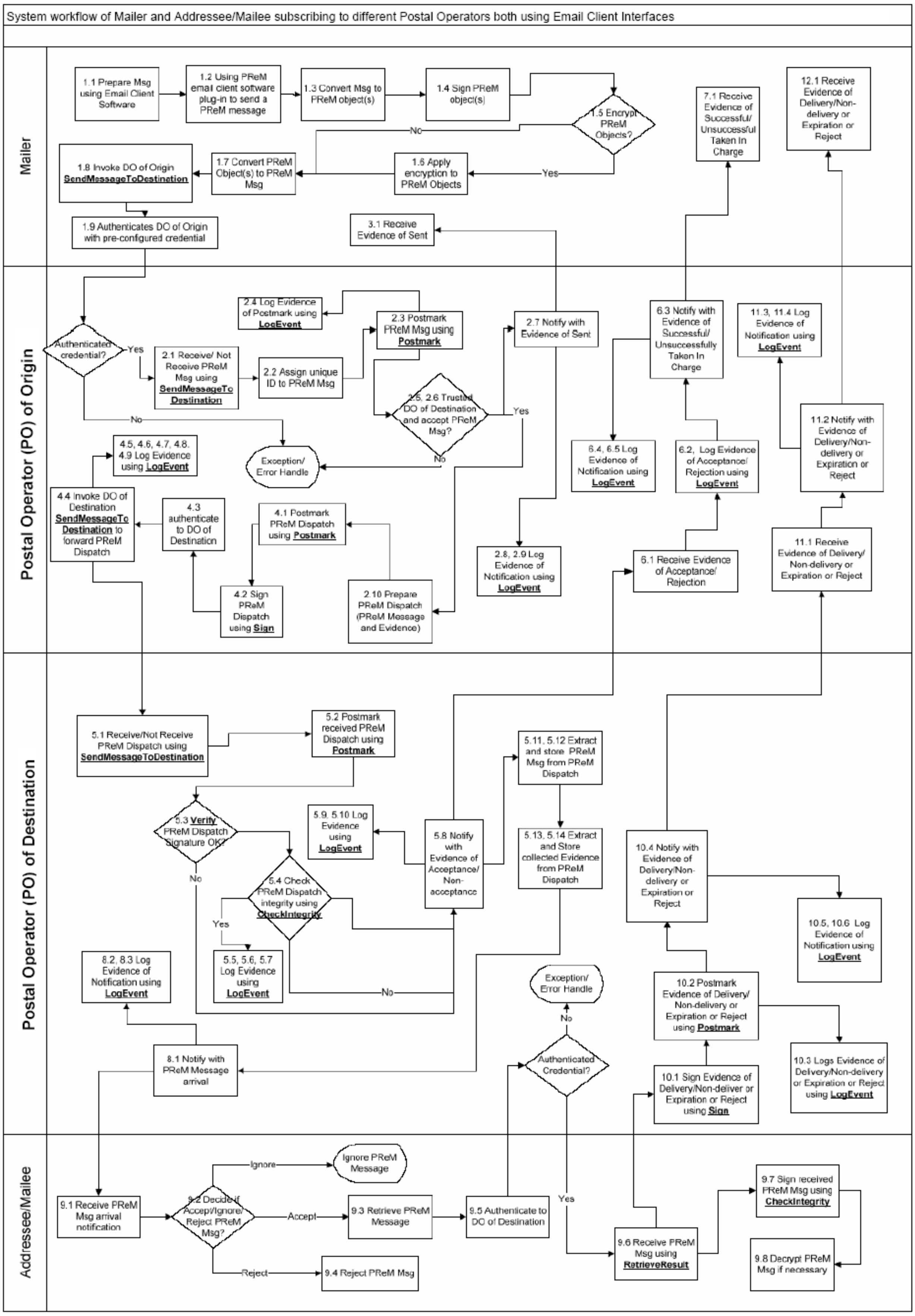
System workflow of Mailer and Addressee/Mailee subscribing to different Postal Operators both using Email Client Interfaces
Mailer
1.1 Prepare Msg using Email Client Software
1.2 Using PReM email client software plug-in to send a PReM message
1.3 Convert Msg to PReM object(s)
1.4 Sign PReM object(s)
1.5 Encrypt PReM Objects?
No
Yes
1.6 Apply encryption to PReM Objects
1.7 Convert PReM Object(s) to PReM Msg
1.8 Invoke DO of Origin SendMessageToDestination
1.9 Authenticates DO of Origin with pre-configured credential
3.1 Receive Evidence of Sent
7.1 Receive Evidence of Successful/ Unsuccessful Taken In Charge
12.1 Receive Evidence of Delivery/Non-delivery or Expiration or Reject
Postal Operator (PO) of Origin
Authenticated credential?
Yes
No
2.1 Receive/ Not Receive PReM Msg using SendMessageTo Destination
2.2 Assign unique ID to PReM Msg
2.3 Postmark PReM Msg using Postmark
2.4 Log Evidence of Postmark using LogEvent
2.5, 2.6 Trusted DO of Destination and accept PReM Msg?
Yes
No
2.7 Notify with Evidence of Sent
2.8, 2.9 Log Evidence of Notification using LogEvent
2.10 Prepare PReM Dispatch (PReM Message and Evidence)
Exception/ Error Handle
4.1 Postmark PReM Dispatch using Postmark
4.2 Sign PReM Dispatch using Sign
4.3 authenticate to DO of Destination
4.4 Invoke DO of Destination SendMessageTo Destination to forward PReM Dispatch
4.5, 4.6, 4.7, 4.8, 4.9 Log Evidence using LogEvent
6.1 Receive Evidence of Acceptance/ Rejection
6.2, Log Evidence of Acceptance/ Rejection using LogEvent
6.3 Notify with Evidence of Successful/ Unsuccessfully Taken In Charge
6.4, 6.5 Log Evidence of Notification using LogEvent
11.1 Receive Evidence of Delivery/ Non-delivery or Expiration or Reject
11.2 Notify with Evidence of Delivery/Non-delivery or Expiration or Reject
11.3, 11.4 Log Evidence of Notification using LogEvent
Postal Operator (PO) of Destination
5.1 Receive/Not Receive PReM Dispatch using SendMessageToDestination
5.2 Postmark received PReM Dispatch using Postmark
5.3 Verify PReM Dispatch Signature OK?
No
5.4 Check PReM Dispatch integrity using CheckIntegrity
Yes
No
5.5, 5.6, 5.7 Log Evidence using LogEvent
5.8 Notify with Evidence of Acceptance/ Non-acceptance
5.9, 5.10 Log Evidence using LogEvent
5.11, 5.12 Extract and store PReM Msg from PReM Dispatch
5.13, 5.14 Extract and Store collected Evidence from PReM Dispatch
8.1 Notify with PReM Message arrival
8.2, 8.3 Log Evidence of Notification using LogEvent
Exception/ Error Handle
No
Authenticated Credential?
Yes
10.1 Sign Evidence of Delivery/Non-deliver or Expiration or Reject using Sign
10.2 Postmark Evidence of Delivery/ Non-delivery or Expiration or Reject using Postmark
10.3 Logs Evidence of Delivery/Non-delivery or Expiration or Reject using LogEvent
10.4 Notify with Evidence of Delivery/Non-delivery or Expiration or Reject
10.5, 10.6 Log Evidence of Notification using LogEvent
Addressee/Mailee
9.1 Receive PReM Msg arrival notification
9.2 Decide if Accept/Ignore/ Reject PReM Msg?
Ignore
Accept
Reject
Ignore PReM Message
9.3 Retrieve PReM Message
9.4 Reject PReM Msg
9.5 Authenticate to DO of Destination
9.6 Receive PReM Msg using RetrieveResult
9.7 Sign received PReM Msg using CheckIntegrity
9.8 Decrypt PReM Msg if necessary

Figure 3 — System workflow of mailer and addressee/mailee subscribing to different postal operators using both Email Client Interface

5.2.2 System workflow of mailer and addressee/mailee using an Email Client Interface and subscribing to PReM Service provided by different postal operators

Actor	Workflow Process
Mailer (Assisted by PReM Email Client Software Plug-in)	1.1 Prepare Message using Email Client Software. 1.2 Use PReM Email Client Software Plug-in to send the PReM Message. 1.3 PReM Email Client Software Plug-in converts Message to PReM object(s). 1.4 PReM Email Client Software Plug-in signs PReM object(s) with mailer's signature private key/password for Proof-of-Ownership and Content Integrity. 1.5 Decide if the PReM object(s) should be encrypted or not with encryption public key of addressee/mailee, which can be searched and retrieved from a public repository located in the postal operator of destination using the Plug-in directory searching feature. 1.6 PReM Email Client Software Plug-in applies encryption to PReM object(s) if necessary. 1.7 PReM Email Client Software Plug-in converts PReM object(s) to PReM Message. 1.8 PReM Email Client Software Plug-in invokes postal operator of origin's **SendMessageToDestination** to send PReM Message to postal operator of origin. 1.9 PReM Email Client Software Plug-in authenticates to a postal operator of origin with a pre-configured credential.
Postal operator of origin	2.1 Receive/Not Receive PReM Message from mailer using **SendMessageToDestination** if authentication of mailer has been Successful/Unsuccessful. 2.2 Assign a unique Identifier to PReM Message. 2.3 Postmark PReM Message using **Postmark** for Proof of Receipt Time from mailer. 2.4 Log "Evidence of Postmark- PReM Message Received – DOO (#26)" into Evidence Store using **LogEvent**. 2.5 Check if postal operator of destination is in the postal operator Trust List. 2.6 Accept/Reject PReM Message from mailer if previous checking has been Successful/Unsuccessful. 2.7 Notify mailer with "Evidence of Sent – PReM Message Acceptance/Rejection – DOO (#01)". 2.8 Log "Evidence of Successful Notification – PReM Message Acceptance/Rejection – DOO (#07)" into Evidence Store using **LogEvent** if notification has been successful. 2.9 Log "Evidence of Failed Notification – PReM Message Acceptance/Rejection – DOO (#08)" into Evidence Store using **LogEvent** if notification has not been successful. 2.10 Prepare PReM Dispatch for postal operator of destination with PReM Message plus all previously collected Evidence.
Mailer	3.1 Receive "Evidence of Sent – PReM Message Acceptance/Rejection – DOO (#01)" from postal operator of origin using Email Client Software.
Postal operator of origin	4.1 Postmark PReM Dispatch using **Postmark** for Proof of Receipt Time by postal operator of origin. 4.2 Sign PReM Dispatch using **Sign** for Proof of Origin to be verified by postal operator of destination. 4.3 Authenticate to postal operator of destination. 4.4 Invoke postal operator of destination's **SendMessageToDestination** to forward mailer's PReM Dispatch to postal operator of destination if authentication has been successful. 4.5 Log "Evidence of Forward – DOO (#03)" using **LogEvent**. 4.6 Log "Evidence of Failed Forward due to Authentication Error – DOO (#05)" using **LogEvent** if authentication has not been successful. 4.7 Log "Evidence of Failed Forward due to Unreachable DOD – DOO (#06)" using **LogEvent** if postal operator of destination cannot be reached. 4.8 Log "Evidence of Failed Forward due to Expiration – DOO (#32)" using **LogEvent** if postal operator of destination does not reply with a correct response within a given time period of time. 4.9 Log "Evidence of Postmark – PReM Dispatch Sent – DOO (#25)", "Evidence of

		Sign – PReM Dispatch – DOO (#02)" into Evidence Store using **LogEvent** if invocation has been successful.
Postal operator of destination	5.1	Receive/Not Receive forwarded PReM Dispatch from postal operator of origin using **SendMessageToDestination** if invocation has been successful/unsuccessful.
	5.2	Postmark received PReM Dispatch using **Postmark** for Proof of Receipt Time by postal operator of destination.
	5.3	Verify PReM Dispatch's Signature of postal operator of origin using **Verify**.
	5.4	Check integrity of PReM Message's Postmark forwarded by postal operator of origin using **CheckIntegrity**.
	5.5	Log "Evidence of Forward – Acceptance/Rejection – DOD (#04)" into Evidence Store using **LogEvent**.
	5.6	Log "Evidence of PReM Dispatch Signature Verification – DOD (#23)" into Evidence Store using **LogEvent**.
	5.7	Log "Evidence of PReM Dispatch Postmark Verification – DOD (#24)" into Evidence Store using **LogEvent**.
	5.8	Notify postal operator of origin with "Evidence of PReM Dispatch – Acceptance/Rejection – DOD (#27)".
	5.9	Log "Evidence of Successful Notification – Acceptance/Rejection – DOD (#09)" into Evidence Store using **LogEvent** if notification has been successful.
	5.10	Log "Evidence of Failed Notification – Acceptance/Rejection – DOD (#10)" into Evidence Store using **LogEvent** if notification has not been successful.
	5.11	Extract PReM Message from PReM Dispatch.
	5.12	Store PReM Message into Message Store.
	5.13	Extract all collected Evidence from PReM Dispatch.
	5.14	Store all collected Evidence into Evidence Store.
Postal operator of origin	6.1	Receive "Evidence of PReM Dispatch – Acceptance/Rejection – DOD (#27)" from postal operator of destination using **ReceiveNotification**.
	6.2	Log "Evidence of PReM Dispatch – Acceptance/Rejection – DOD (#27)" into Evidence Store using **LogEvent**.
	6.3	Notify mailer with "Evidence of Successfully/Unsuccessfully Taken in Charge – DOD (#28)".
	6.4	Log "Evidence of Successful Notification – Successfully/Unsuccessfully Taken in Charge – DOD (#11)" into Evidence Store using **LogEvent** if notification has been successful.
	6.5	Log "Evidence of Failed Notification – Successfully/Unsuccessfully Taken in Charge DOD (#12)" into Evidence Store using **LogEvent** if notification has not been successful.
Mailer	7.1	Receive "Evidence of Successful/Unsuccessful Taken in Charge – DOD (#28)" using Email Client Software.
Postal operator of destination	8.1	Notify addressee/mailee with "Evidence of PReM Dispatch – Acceptance/Rejection – DOD (#27)", informing that PReM Message is available to be received.
	8.2	Log "Evidence of Successful Notification – PReM Message Arrival – DOD (#13)" into Evidence Store using **LogEvent** if notification has been successful.
	8.3	Log "Evidence of Failed Notification – PReM Message Arrival – DOD (#14)" into Evidence Store using **LogEvent** if notification has not been successful.
Addressee/mailee (Assisted by PReM Email Client Software Plug-in)	9.1	Receive PReM Message Arrival Notification from postal operator of destination using Email Client Software.
	9.2	Decide if PReM Message from mailer should be accepted/ignored/rejected.
	9.3	Use/Not Use PReM Email Client Software Plug-in to retrieve PReM Message if addressee/mailee accepts/ignores PReM Message.
	9.4	Reject PReM Message with PReM Email Client Software Plug-in using **RejectMessage** if addressee/mailee rejects PReM Message.
	9.5	PReM Email Client Software Plug-in authenticates to postal operator of destination with the pre-configured credential in PReM Email Client Software Plug-in.
	9.6	PReM Email Client Software Plug-in receives/does not receive PReM Message using **RetrieveResults** if authentication of addressee/mailee has been successful/ unsuccessful.
	9.7	PReM Email Client Software Plug-in signs received PReM Message with addressee/mailee's signature private key/password using **CheckIntegrity** for Proof-

	of- Possession. 9.8 PReM Email Client Software Plug-in decrypts PReM Message using addressee/mailee's encryption private key if it is encrypted.
Postal operator of destination	10.1 Sign "Evidence of Delivery/Non-delivery – addressee/mailee (#29)" or "Evidence of PReM Message Expiration – addressee/mailee (#30)" or "Evidence of Reject – addressee/mailee (#31)" using **Sign** depending on whether addressee/mailee accepts/ignores/rejects PReM Message. 10.2 Postmark "Evidence of Delivery/Non-delivery – addressee/mailee (#29)" or "Evidence of PReM Message Expiration – addressee/mailee (#30)" or "Evidence of Reject – addressee/mailee (#31)" using **Postmark** for Proof-of-Reception/Non-reception by addressee/mailee. 10.3 Log "Evidence of Delivery/Non-delivery-addressee/mailee (#29)" or "Evidence of PReM Message Expiration – addressee/mailee (#30)" or "Evidence of Reject – addressee/mailee (#31)" into Evidence Store using **LogEvent**. 10.4 Notify postal operator of origin with "Evidence of Delivery/Non-delivery – addressee/mailee (#29)" or "Evidence of PReM Message Expiration – addressee/mailee (#30)" or "Evidence of Reject – addressee/mailee (#31)", informing successful/unsuccessful completion of the Receive operation. 10.5 Log "Evidence of Successful Notification – Delivery/Non-delivery – DOD (#15)" or "Evidence of Successful Notification – PReM Message Expiration – DOD (#19)" or "Evidence of Successful Notification – PReM Message Rejected – DOD (#09)" into Evidence Store using **LogEvent** if notification has been successful. 10.6 Log "Evidence of Failed Notification – Delivery/Non-delivery – DOD (#16)" or "Evidence of Failed Notification – PReM Message Expiration – DOD (#20)" or "Evidence of Failed Notification – Acceptance/Rejection – DOD (#10)" into evidence Store using **LogEvent** if notification has not been successful.
Postal operator of origin	11.1 Receive "Evidence of Delivery/Non-delivery – addressee/mailee (#29)" or "Evidence of PReM Message Expiration – addressee/mailee (#30)" or "Evidence of Reject – addressee/mailee (#31)" from postal operator of destination. 11.2 Notify mailer with "Evidence of Delivery/Non-delivery – addressee/mailee (#29)" or "Evidence of PReM Message Expiration – addressee/mailee (#30)" or "Evidence of Reject – addressee/mailee (#31)" informing that his/her PReM Message has been successfully/unsuccessfully received or ignored or rejected by the addressee/mailee. 11.3 Log "Evidence of Successful Notification – Delivery/Non-delivery – DOO (#17)" or "Evidence of Successful Notification – PReM Message Expiration – DOO (#21)" or "Evidence of Successful Notification – PReM Message Acceptance/Rejection – DOO (#07)" to Evidence Store using **LogEvent** if notification has been successful. 11.4 Log "Evidence of Failed Notification – Delivery/Non-delivery – DOO (#18)" or "Evidence of Failed Notification – PReM Message Expiration – DOO (#22)" or "Evidence of Failed Notification – PReM Message Acceptance/Rejection – DOO (#08)" to Evidence Store using **LogEvent** if notification has not been successful.
Mailer	12.1 Receive "Evidence of Delivery/Non-delivery – addressee/mailee (#29)" or "Evidence of PReM Message Expiration – addressee/mailee (#30)" or "Evidence of Reject – addressee/mailee (#31)" from postal operator of origin using Email Client Software, proving that addressee/mailee has received/rejected/ignore his/her PReM Message.

5.2.3 Mailer and addressee/mailee both using Web-based Interface and subscribing to PReM Service provided by different postal operators

The PReM Web-based Interface operates under the assumption that the mailer and addressee/mailee use standard web browser software. Examples of such software include, but are not limited to, Microsoft Internet Explorer, Mozilla Firefox, Opera, etc. The PReM End User will need to install software modules for authentication purposes, but should not need to install any other additional software.

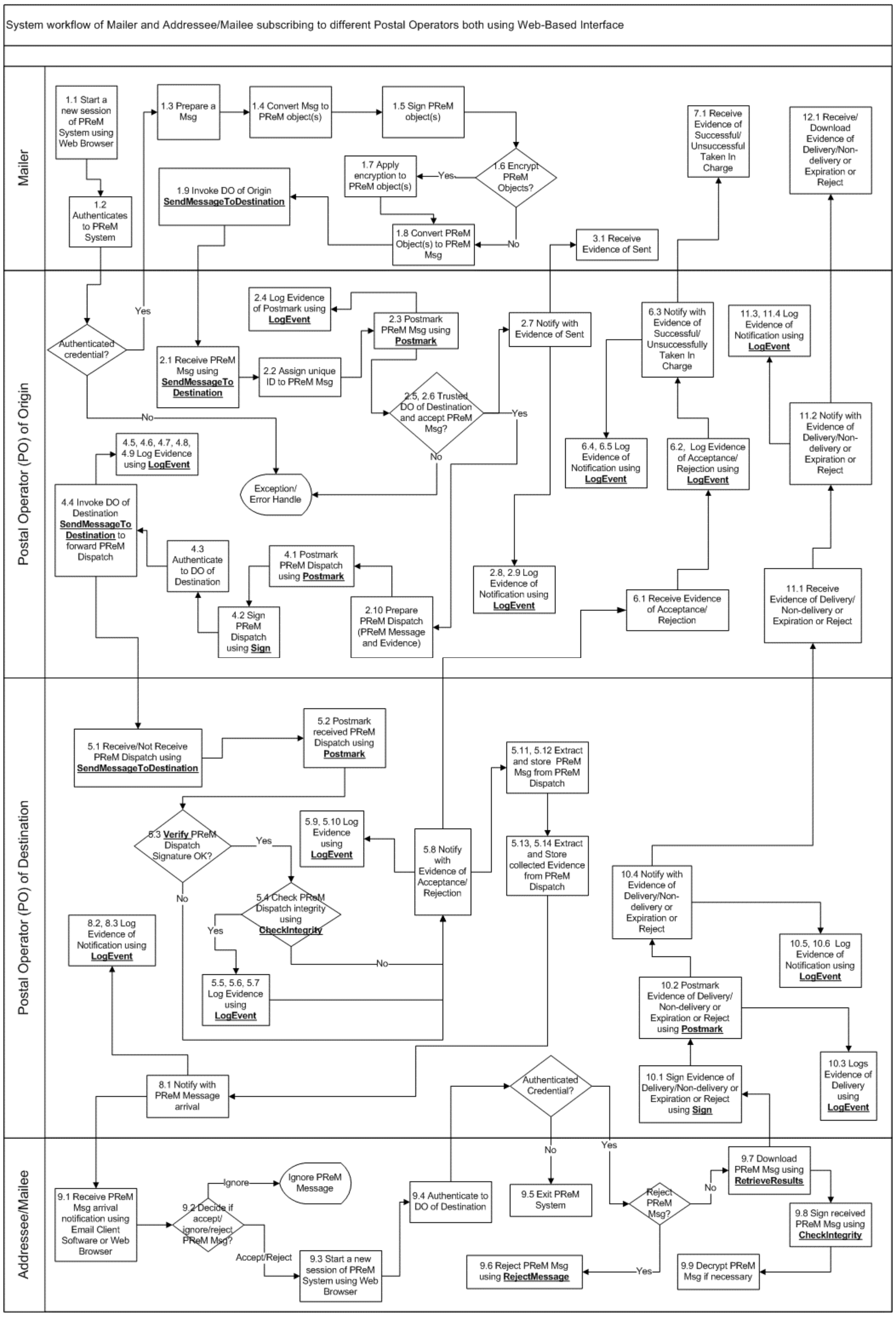
System workflow of Mailer and Addressee/Mailee subscribing to different Postal Operators both using Web-Based Interface
Mailer
Postal Operator (PO) of Origin
Postal Operator (PO) of Destination
Addressee/Mailee
1.1 Start a new session of PReM System using Web Browser
1.2 Authenticates to PReM System
1.3 Prepare a Msg
1.4 Convert Msg to PReM object(s)
1.5 Sign PReM object(s)
1.6 Encrypt PReM Objects?
1.7 Apply encryption to PReM object(s)
1.8 Convert PReM Object(s) to PReM Msg
1.9 Invoke DO of Origin SendMessageToDestination
Authenticated credential?
2.1 Receive PReM Msg using SendMessageTo Destination
2.2 Assign unique ID to PReM Msg
2.3 Postmark PReM Msg using Postmark
2.4 Log Evidence of Postmark using LogEvent
2.5, 2.6 Trusted DO of Destination and accept PReM Msg?
2.7 Notify with Evidence of Sent
2.8, 2.9 Log Evidence of Notification using LogEvent
2.10 Prepare PReM Dispatch (PReM Message and Evidence)
3.1 Receive Evidence of Sent
Exception/ Error Handle
4.1 Postmark PReM Dispatch using Postmark
4.2 Sign PReM Dispatch using Sign
4.3 Authenticate to DO of Destination
4.4 Invoke DO of Destination SendMessageTo Destination to forward PReM Dispatch
4.5, 4.6, 4.7, 4.8, 4.9 Log Evidence using LogEvent
5.1 Receive/Not Receive PReM Dispatch using SendMessageToDestination
5.2 Postmark received PReM Dispatch using Postmark
5.3 Verify PReM Dispatch Signature OK?
5.4 Check PReM Dispatch integrity using CheckIntegrity
5.5, 5.6, 5.7 Log Evidence using LogEvent
5.8 Notify with Evidence of Acceptance/ Rejection
5.9, 5.10 Log Evidence using LogEvent
5.11, 5.12 Extract and store PReM Msg from PReM Dispatch
5.13, 5.14 Extract and Store collected Evidence from PReM Dispatch
6.1 Receive Evidence of Acceptance/ Rejection
6.2, Log Evidence of Acceptance/ Rejection using LogEvent
6.3 Notify with Evidence of Successful/ Unsuccessfully Taken In Charge
6.4, 6.5 Log Evidence of Notification using LogEvent
7.1 Receive Evidence of Successful/ Unsuccessful Taken In Charge
8.1 Notify with PReM Message arrival
8.2, 8.3 Log Evidence of Notification using LogEvent
9.1 Receive PReM Msg arrival notification using Email Client Software or Web Browser
9.2 Decide if accept/ ignore/reject PReM Msg?
Ignore PReM Message
9.3 Start a new session of PReM System using Web Browser
9.4 Authenticate to DO of Destination
Authenticated Credential?
9.5 Exit PReM System
Reject PReM Msg?
9.6 Reject PReM Msg using RejectMessage
9.7 Download PReM Msg using RetrieveResults
9.8 Sign received PReM Msg using CheckIntegrity
9.9 Decrypt PReM Msg if necessary
10.1 Sign Evidence of Delivery/Non-delivery or Expiration or Reject using Sign
10.2 Postmark Evidence of Delivery/ Non-delivery or Expiration or Reject using Postmark
10.3 Logs Evidence of Delivery using LogEvent
10.4 Notify with Evidence of Delivery/Non-delivery or Expiration or Reject
10.5, 10.6 Log Evidence of Notification using LogEvent
11.1 Receive Evidence of Delivery/ Non-delivery or Expiration or Reject
11.2 Notify with Evidence of Delivery/Non-delivery or Expiration or Reject
11.3, 11.4 Log Evidence of Notification using LogEvent
12.1 Receive/ Download Evidence of Delivery/Non-delivery or Expiration or Reject
Yes
No
Ignore
Accept/Reject

Figure 4 — System workflow of mailer and addressee/mailee subscribing to different postal operators both using Web-based Interface

5.2.4 System Workflow of mailer and addressee/mailee using Web-based Interfaces and subscribing to PReM Service provided by different postal operators.

Actor	Workflow Process
Mailer (Assisted by PReM Email Client Software Plug-in)	1.1 Start a new session of PReM System using Web Browser. 1.2 Authenticate to PReM System using mailer's identity private key/password. 1.3 Prepare Message/Exit if authentication has been successful/unsuccessful. 1.4 Web browser converts Message to PReM object(s). 1.5 Web browser signs PReM object(s) using mailer's signature private key/password for Proof-of-Ownership and Content Integrity. 1.6 Decide if Message should be encrypted or not with encryption public key of addressee/mailee, which can be searched and retrieved from a public repository located in the postal operator of destination using the PReM System's directory searching feature. 1.7 Web browser applies encryption to PReM object(s) if necessary. 1.8 Web Browser converts PReM object(s) to PReM Message. 1.9 Web Browser invokes the postal operator of origin's **SendMessageToDestination** to upload PReM Message to postal operator of origin.
Postal operator of origin	2.1 Receive PReM Message from mailer using **SendMessageToDestination**. 2.2 Assign a unique Identifier to PReM Message. 2.3 Postmark PReM Message using **Postmark**. 2.4 Log "Evidence of Postmark – PReM Message Received – DOO (#26)" into Evidence Store using **LogEvent**. 2.5 Check if postal operator of destination is in the postal operator Trust List. 2.6 Accept/Reject PReM Message from mailer if previous checking has been Successful/Unsuccessful. 2.7 Notify mailer with "Evidence of Sent – PReM Message Acceptance/Rejection – DOO (#01)". 2.8 Log "Evidence of Successful Notification – PReM Message Acceptance/Rejection – DOO (#07)" into Evidence Store using **LogEvent** if notification has been successful. 2.9 Log "Evidence of Failed Notification – PReM Message Acceptance/Rejection – DOO (#08)" into Evidence Store using **LogEvent** if notification has not been successful. 2.10 Prepare PReM Dispatch for postal operator of destination with PReM Message plus all previously collected evidence.
Mailer	3.1 Receive "Evidence of Sent – PReM Message Acceptance/Rejection – DOO (#01)" from postal operator of origin using Email Client Software/Web Browser.
Postal operator of origin	4.1 Postmark PReM Dispatch using **Postmark** for Proof of Receipt Time by postal operator of origin. 4.2 Sign PReM Dispatch using **Sign** for Proof of Origin. 4.3 Authenticate to postal operator Of Destination. 4.4 Invoke postal operator of destination's **SendMessageToDestination** to forward mailer's PReM Dispatch to postal operator of destination if authentication has been successful. 4.5 Log "Evidence of Forward – DOO (#03)" using **LogEvent**. 4.6 Log "Evidence of Failed Forward due to Authentication Error – DOO (#05)" using **LogEvent** if authentication has not been successful. 4.7 Log "Evidence of Failed Forward due to Unreachable DOD – DOO (#06)" using **LogEvent** if postal operator of destination cannot be reached. 4.8 Log "Evidence of Failed Forward due to Expiration – DOO (#32)" using **LogEvent** if postal operator of destination does not reply with a correct response within a given time period of time. 4.9 Log "Evidence of Postmark- PReM Dispatch Sent – DOO (#25)", "Evidence of Sign – PReM Dispatch – DOO (#02)" into Evidence Store using **LogEvent** if invocation has been successful.
Postal operator of destination	5.1 Receive/Not Receive forwarded PReM Dispatch from postal operator of origin using **SendMessageToDestination** if invocation has been successful/unsuccessful. 5.2 Postmark received PReM Dispatch using **Postmark** for Proof of Receipt Time by postal operator of destination. 5.3 Verify PReM Dispatch's Signature using **Verify**.

	5.4 Check integrity of PReM Message's Postmark forwarded by postal operator of origin using **CheckIntegrity**. 5.5 Log "Evidence of Forward – Acceptance/Rejection – DOD (#04)" into Evidence Store using **LogEvent**. 5.6 Log "Evidence of PReM Dispatch Signature Verification – DOD (#23)" into Evidence Store using **LogEvent**. 5.7 Log "Evidence of PReM Dispatch Postmark Verification – DOD (#24)" into Evidence Store using **LogEvent**. 5.8 Notify postal operator of origin with "Evidence of PReM Dispatch – Acceptance/Rejection – DOD (#27)". 5.9 Log "Evidence of Successful Notification – Acceptance/Rejection -DOD (#09)" into Evidence Store using **LogEvent** if notification has been successful. 5.10 Log "Evidence of Failed Notification – Acceptance/Rejection – DOD (#10)" into Evidence Store using **LogEvent** if notification has not been successful. 5.11 Extract PReM Message from PReM Dispatch. 5.12 Store PReM Message into Message Store. 5.13 Extract all collected Evidence from PReM Dispatch. 5.14 Store all collected Evidence into Evidence Store.
Postal operator of origin	6.1 Receive "Evidence of PReM Dispatch – Acceptance/Rejection – DOD (#27)" from postal operator of destination using **ReceiveNotification**. 6.2 Log "Evidence of PReM Dispatch – Acceptance/Rejection – DOD (#27)" into Evidence Store using **LogEvent**. 6.3 Notify mailer with "Evidence of Successfully/Unsuccessfully Taken in Charge – DOD (#28)". 6.4 Log "Evidence of Successful Notification – Successfully/Unsuccessfully Taken in Charge – DOD (#11)" into Evidence Store using **LogEvent** if notification has been successful. 6.5 Log "Evidence of Failed Notification – Successfully/Unsuccessfully Taken in Charge – DOD (#12)" into Evidence Store using **LogEvent** if notification has not been successful.
Mailer	7.1 Receive/Download "Evidence of Successful/Unsuccessful Taken in Charge – DOD (#28)" using Email Client Software/Web Browser.
Postal operator of destination	8.1 Notify addressee/mailee with "Evidence of PReM Dispatch – Acceptance/Rejection – DOD (#27)", informing that PReM Message is available to receive. 8.2 Log "Evidence of Successful Notification – PReM Message Arrival – DOD (#13)" into Evidence Store using **LogEvent** if notification has been successful. 8.3 Log "Evidence of Failed Notification – PReM Message Arrival – DOD (#14)" into Evidence Store using **LogEvent** if notification has not been successful.
Addressee/mailee (Assisted by PReM Email Client Software Plug-in)	9.1 Receive/Download PReM Message Arrival Notification from postal operator of destination using Email Client Software/Web Browser. 9.2 Decide if PReM Message from mailer should be accepted/ignored/rejected. 9.3 Start a new session of PReM System using Web Browser if addressee/mailee does not ignore PReM Message. 9.4 Authenticate to postal operator of destination with addressee/mailee's identity private key/password. 9.5 Exit PReM System if the authentication has not been successful. 9.6 Reject PReM Message with Web Browser using **RejectMessage** if addressee/mailee rejects PReM Message. 9.7 Download/Not download PReM Message with Web Browser using **RetrieveResults** if addressee/mailee accepts/rejects PReM Message. 9.8 Sign received PReM Message with addressee/mailee's signature private key/password using **CheckIntegrity** for Proof-of-Possession. 9.9 Web Browser decrypts PReM Message using addressee/mailee's encryption private key if it is encrypted.
Postal operator of destination	10.1 Sign "Evidence of Delivery/Non-delivery-addressee/mailee (#29)" or "Evidence of PReM Message Expiration – addressee/mailee (#30)" or "Evidence of Reject – addressee/mailee (#31)" using **Sign** depending on whether addressee/mailee accepts/ignores/rejects PReM Message. 10.2 Postmark "Evidence of Delivery/Non-delivery – addressee/mailee (#29)" or Evidence

	of PReM Message Expiration – addressee/mailee (#30)” or “Evidence of Reject – addressee/mailee (#31)” using **Postmark**. 10.3 Log “Evidence of Delivery/Non-delivery – addressee/mailee (#29)” or “Evidence of PReM Message Expiration – addressee/mailee (#30)” or “Evidence of Reject – addressee/mailee (#31)” into the Evidence Store using **LogEvent**. 10.4 Notify postal operator of origin with “Evidence of Delivery/Non-delivery – addressee/mailee (#29)” or “Evidence of PReM Message Expiration – addressee/mailee (#30)” or “Evidence of Reject – addressee/mailee (#31)”, informing successful/unsuccessful completion of the Receive operation. 10.5 Log “Evidence of Successful Notification – Delivery/Non-delivery – DOD (#15)” or “Evidence of Successful Notification – PReM Message Expiration – DOD (#19)” or “Evidence of Successful Notification – PReM Message Rejected – DOD (#09)” into Evidence Store using **LogEvent** if notification has been successful. 10.6 Log “Evidence of Failed Notification – Delivery/Non-delivery – DOD (#16)” or “Evidence of Failed Notification – PReM Message Expiration – DOD (#20)” or “Evidence of Failed Notification – Acceptance/Rejection – DOD (#10)” into Evidence Store using **LogEvent** if notification has not been successful.
Postal operator of origin	11.1 Receive “Evidence of Delivery/Non-delivery – addressee/mailee (#29)” or “Evidence of PReM Message Expiration – addressee/mailee (#30)” or “Evidence of Reject – addressee/mailee (#31)” from postal operator of destination. 11.2 Notify mailer with “Evidence of Delivery/Non-delivery – addressee/mailee (#29)” or “Evidence of PReM Message Expiration – addressee/mailee (#30)” or “Evidence of Reject – addressee/mailee (#31)” informing that his/her PReM Message has been successfully/unsuccessfully received or ignored or rejected by the addressee/mailee. 11.3 Log “Evidence of Successful Notification – Delivery/Non-delivery – DOO (#18)” or “Evidence of Successful Notification – PReM Message Expiration – DOO (#21)” or “Evidence of Successful Notification – PReM Message Acceptance/Rejection – DOO (#07)” to Evidence Store using **LogEvent** if notification has been successful. 11.4 Log “Evidence of Failed Notification – Delivery/Non-delivery – DOO (#18)” or “Evidence of Failed Notification – PReM Message Expiration – DOO (#22)” or “Evidence of Failed Notification – PReM Message Acceptance/Rejection – DOO (#08)” to Evidence Store using **LogEvent** if notification has not been successful.
Mailer	12.1 Receive/Download “Evidence of Delivery/Non-delivery – addressee/mailee (#29)” or “Evidence of PReM Message Expiration – addressee/mailee (#30)” or “Evidence of Reject – addressee/mailee (#31)” from postal operator of origin using Email Client Software/Web Browser, proving that addressee/mailee has received/rejected his/her PReM Message.

5.2.5 PReM implementation

When implementing a PReM service, postal operators should provide PReM End Users the ability to use either Email Client Interface and/or Web-based Interface.

In addition, Web Service is recommended for postal operators as the implementation mechanism for PReM Dispatch transmission and evidence relay (verbs) among themselves.

Allthough the postal operators should generate and store all the evidence types that are mentioned in 8.2, only those defined as mandatory should be sent to the PReM End User.

5.2.6 Interoperation between PReM System and non-PReM System

PReM Dispatch or Evidence transmission between a PReM System and a non-PReM System is not guaranteed due to the fact that necessary verbs responsible for PReM Dispatch and Evidence transmission are not available in a non-PReM System. It is permissible, though not required, for postal operators to implement an interface between their PReM system and electronic communications systems that are not PReM compliant. Where this is done a mailer sending a message which will leave the PReM system domain shall be clearly warned of this fact and asked to confirm approval; similarly, the addressee/mailee of a message which entered from outside the PReM domain shall be clearly advised of this fact.

6 Roles in PReM

Within the scope of a PReM System, there are a number of roles which contribute their importance to the system operations and workflows. They are listed as follows:

— **addressee**: party who is the intended ultimate recipient of a PReM message.

— **authorised parties**: party who is interested in the evidence of the PReM operation and is authorised by the postal operators to retrieve evidence stored in the evidence store.

— **postal operator of origin**: postal operator responsible for the acceptance of PReM Message from mailer and the management of the outgoing PReM Message. It is also considered as a Management Domain.

— **postal operator of destination**: postal operator responsible for the delivery of PReM Message to addressee and the management of the incoming PReM Message. It is also considered as a Management Domain.

— **postal operator trust list distribution point**: party responsible for the management of a list of authenticated postal operators which are authorised to provide the PReM service into a trust list such that users of the PReM System will only be allowed to send PReM Message to the postal operators inside the trust list.

— **evidence store**: repository where PReM operation evidence is stored.

— **mailee**: party having responsibility for ensuring that a PReM Message delivered or handed over by the postal operator of destination at the delivery address, reaches the addressee.

— **mailer**: party who carries out one or more of the processes involved in creating, sending and paying the postage due for a PreM message.

— **message store**: repository where PReM messages are stored.

— **PReM policy domain**: any domain where a common set of rules (e.g. legal, company policy or agreement) is enforced for the provision of PReM services. This domain may be, but is not limited to, a country, a company, an organisation and a group of countries.

— **PReM end user repository**: a public directory service where user information of mailers, addressees and mailees is stored, public search or locate operation is allowed and enabled with searching criteria such as, but not limited to, email address, common name, first name and last name.

— **management domain**: set of technical and physical components, personnel, policies and processes which provides Registered Electronic Mail (REM) services and is operated under the responsibility of a single management entity.

7 Functional specification

7.1 Introduction

This clause defines the verbs used to control operation of the PReM system. Although all required verbs are listed, many are defined in CEN/TS 15121-1:2011. The subclauses covering these are limited to a cross-reference to CEN/TS 15121-1:2011, the CEN standard in which SePS is defined.

Detailed descriptions are given of five verbs which are specific to PReM but, even for these, most of the concepts, request elements, option flags, etc, are defined in UPU S43a and the reader is referred to that standard for relevant detail.

7.2 Functional description

7.2.1 CheckIntegrity

See CEN/TS 15121-1:2011.

7.2.2 LogEvent

See CEN/TS 15121-1:2011.

7.2.3 PostMark

See CEN/TS 15121-1:2011.

7.2.4 RetrieveResults

See CEN/TS 15121-1:2011.

7.2.5 Sign

See CEN/TS 15121-1:2011.

7.2.6 Verify

See CEN/TS 15121-1:2011.

7.2.7 SendMessageToDestination

7.2.7.1 General

The SendMessageToDestination operation will send a PReM message to a given `Destination`, and optionally returns a `PostMarkedReceipt` as proof of receiving successfully the PReM message by the callee. In this case, the callee can be either the postal operator of origin or the postal operator of destination.

SendMessageToDestination operates in two modes, depending on whether delivery is local or remote. A delivery is local when the PReM System which is performing this operation is the postal operator of destination, whereas a delivery is considered as remote when the PReM System which is executing this operation is **not** the postal operator of destination.

For remote delivery, the postal operator of origin shall use its best efforts to forward the PReM message to the postal operator of destination. In this case, the postal operator of origin, while performing the

SendMessageToDestination operation called by the mailer, should issue its own SendMessageToDestination operation to the postal operator of destination.

In the absence of any other means of authentication, the system can determine whether the PReM message is a normal message or a forwarded message from another Operator by checking the `ClaimedIdentity` in the SendMessageToDestinationRequest.

The SendMessageToDestination operation is a high-level compound operation which depends on other low-level operations defined in the SePS interface specification such as **Postmark**, **Sign**, **LogEvent** and **CheckIntegrity**.

7.2.7.2 SendMessageToDestination Request Flags

The following option flags can be set in the `SendMessageToDestinationOptionsType` complex element. These elements are of type Boolean and should be set to ‘True’ or ‘False’. The default value is ‘False’.

```
<xs:complexType name=”SendMessageToDestinationOptionsType”>
     <xs:sequence>
          <xs:element name=”EndLifeCycle” type=”xs:boolean”/>
          <xs:element name=”ExtendLifeCycle” type=”xs:boolean”/>
          <xs:element name=”IssuePostMarkedReceipt” type=”xs:boolean” />
     </xs:sequence>
</xs:complexType>
```

EndLifeCycle – Set to ‘True’ to end the transaction Lifecycle (see CEN/TS 15121-1:2011).

ExtendLifeCycle – Set to ‘True’ to extend the transaction lifecycle (see CEN/TS 15121-1:2011).

IssuePostMarkedReceipt – Set to ‘True’ to receive a receipt to attest that the message has been successfully received by the PReM System. Details of the `PostMarkedReceipt` complex element are given in CEN/TS 15121-1:2011.

7.2.7.3 SendMessageToDestination Request Element

```
<xs:element name=”SendMessageToDestinationRequest”>
     <xs:complexType>
          <xs:sequence>
               <xs:element name=”SendMessageToDestinationOptions”
type=”prem:SendMessageToDestinationOptionsType”/>
               <xs:element name=”TransactionKey” type=”epm:TransactionKeyType” />
          <xs:element name=”OriginalClaimedIdentity” type=”epm:ClaimedIdentityType” />
               <xs:element name=”ClaimedIdentity” type=”epm:ClaimedIdentityType” />
               <xs:element name=”OrganizationID” type=”xs:string” nillable=”true”/>
               <xs:element name=”ClientApplication”type=”epm:ClientApplicationType” />
               <xs:element name=”ContentIdentifier” type=”xs:string” nillable=”true” />
               <xs:element name=”Destination” type=”xs:string” />
               <xs:element name=”Timeout” type=”xs:integer”/>
               <xs:element name=”Data” type=”epm:QualifiedDataType” />
               <xs:element name=”ContentMetadata” type=”epm:ContentMetadataType”
nillable=”true” minOccurs=”0” maxOccurs=”unbounded” />
          </xs:sequence>
     </xs:complexType>
</xs:element>
```

TransactionKey – An element whose type is a complex element defined by `TransactionKeyType` (see CEN/TS 15121-1:2011).

OriginalClaimedIdentity – This element is to identify the Sender of the PReM System and shall be initialised for forwarded messages. If the Sender of the message did not initialise ClaimedIdentity, it is the responsibility of the postal operator of origin to initialise this element before forwarding the PReM Message to the postal operator of destination.

ClaimedIdentity – The RequesterSignature element of ClaimedIdentity shall be initialised if the caller is not previously authenticated.

OrganisationID – A string element which identifies the organisation requesting this service. This value shall match against a list of valid Organisation Identifiers which are registered with the PReM System.

ClientApplication – see CEN/TS 15121-1:2011 for details of the `ClientApplicationType` complex element.

ContentIdentifier – This is an optional element which can be used to provide an application-specific identifier for the type of document being sent. See CEN/TS 15121-1:2011 for details of the possible usage of this element.

Destination –The `Destination` is a string which follows the RFC 2822 [1] (Internet Message Format) convention, consisting of two parts: local part and domain part. The domain part determines the postal operator of destination. The implementation of the PReM System is free to choose its own way of dealing with the local part, as long as it conforms to the RFC 2822 [1] addressing conventions.

Data – A binary element initialised by the caller which embodies the PReM message to be sent to the destination.

Timeout – This element specifies the period of time (in hours) that the PReM system should wait before considering a PReM Message as **not received** by the mailee/addressee. Set to 0 for the default timeout value, which is determined by the postal operator of destination. The postal operator of destination **may** choose to ignore this parameter and to apply the default value instead.

ContentMetadata – A complex element which describes the custom details of the PReM to be sent. See CEN/TS 15121-1:2011 for information on its possible usage.

7.2.7.4 SendMessageToDestination Response Objects

```
<xs:element name="SendMessageToDestinationResponse">
     <xs:complexType>
          <xs:element name="TransactionStatus" type="xs:string"/>
          <xs:element name="TransactionStatusDetail"
          type="epm:TransactionStatusDetailType"/>
          <xs:element name="TransactionKey" type="epm:TransactionKeyType"/>
          <xs:element name="PostMarkedReceipt" type="epm:PostMarkedReceiptType"
nillable="true"/>
     </xs:complexType>
</xs:element>
```

TransactionStatus – A string element representing the result of the overall transaction. ‘0’ denotes success. ‘1’ denotes that a warning is generated. Transaction failure is denoted by a specific numerical value.

TransactionStatusDetail – An element whose type is a complex element defined by `TransactionStatusDetailType`. See UPU S43a for details of both TransactionStatus and TransactionStatus Detail.

TransactionKey – An element whose type is a complex element defined by `TransactionKeyType` (see CEN/TS 15121-1:2011) A new transaction key is returned if the caller does not specify one (implicit start of a new transaction).

PostMarkedReceipt – The `PostMarkedReceipt` element (see CEN/TS 15121-1:2011) is returned if the IssuePostMarkedReceipt flag of the SendMessageToDestinationRequest element is set to ‘True’. The PReM System will issue a postmark that attests the time and the integrity of the PReM passed by the caller.

7.2.8 RejectMessage

7.2.8.1 General

RejectMessage is called to explicitly reject the reception of a PReM message by the addressee/mailee.

NOTE The reception of a PReM Message can also be implicitly cancelled by not retrieving it: after a certain period of time of non-retrieval (timeout), the PReM Message is considered as expired due to non-retrieval. For further details, see the description of the Timeout parameter in 7.2.7.3 above.

7.2.8.2 RejectMessage Request Element

```
<xs:element name=”RejectMessageRequest”>
     <xs:complexType>
          <xs:sequence>
               <xs:element name=”TransactionKey” type=”epm:TransactionKeyType” />
          </xs:sequence>
     </xs:complexType>
</xs:element>
```

TransactionKey – An element whose type is a complex element defined by `TransactionKeyType` (see CEN/TS 15121-1:2011) to describe the PReM Message sent using `SendMessageToDestination`. In other words, the notification message to the addressee/mailee should include the TransactionKey or some other keys which are able to retrieve the original Transaction Key.

7.2.8.3 RejectMessage Response Element

```
<xs:element name=”SendMessageToDestinationResponse”>
     <xs:complexType>
          <xs:element name=”TransactionStatus” type=”xs:string”/>
          <xs:element name=”TransactionStatusDetail”
          type=”epm:TransactionStatusDetailType”/>
          <xs:element name=”TransactionKey” type=”epm:TransactionKeyType”/>
          <xs:element name=”PostMarkedReceipt” type=”epm:PostMarkedReceiptType”
nillable=”true”/>
     </xs:complexType>
</xs:element>
```

TransactionStatus – A string element representing the result of the overall transaction. ‘0’ denotes success. ‘1’ denotes that a warning is generated. Transaction failure is denoted by a specific numerical value.

TransactionStatusDetail – An element whose type is a complex element defined by `TransactionStatus DetailType`. See CEN/TS 15121-1:2011 for details related to TransactionStatus and TransactionStatus Detail.

TransactionKey – An element whose type is a complex element defined by `TransactionKeyType` (see CEN/TS 15121-1:2011).

7.2.9 SubscribeNotification

7.2.9.1 General

The PReM System supports a flexible and dynamic notification scheme to which the components or parties interested in certain PReM events can subscribe and they will be notified via a callback function. The caller can, through this interface, implement different types of notification to the PReM end users, such as sending SMS, sending an ordinary email notification, or performing some administrative and logging functions.

7.2.9.2 SubscribeNotification Request Element

```
<xs:element name = "SubscribeNotificationRequest">
     <xs:complexType>
          <xs:sequence>
               <xs:element name="EventType" type="xsd:string"/>
               <xs:element name="ClientApplication" type="epm:ClientApplicationType" />
               <xs:element name="CallbackUrl" type="xsd:string"/>
          </xs:sequence>
     </xs:complexType>
</xs:element>
```

EventType – A string element which denotes the event of which the caller intends to receive notification through a callback function. The events include, but are not limited to:

- **MessageAccepted**: the message was accepted by the postal operator of origin/destination

- **MessageRejected**: the message was rejected by the postal operator of origin/destination

- **MessageDeliveryFailure**: the message could not be delivered to the postal operator of destination or mailee/addressee, e.g. network failure

- **MessageReceivedFromOrigin**: the message was successfully received from the postal operator of origin and ready to be sent to the addressee/mailee

- **MessageAcceptedByaddressee**: the message was successfully received by the addressee/mailee

- **MessageRejectedByaddressee**: the message was explicitly rejected by the addressee/mailee

- **MessageExpired**: the message was not retrieved by the addressee/mailee after a certain period of time **ClientApplication**: a complex element describing the caller who intends to subscribe to the event. See CEN/TS 15121-1:2011 for details of the `ClientApplicationType` complex element.

- **CallbackUrl**: the URL of the callback function which the PReM System will call after the event denoted by `EventType` happens. The callback function must adhere to the interface `ReceiveNotification` (see 7.2.11).

7.2.9.3 SubscribeNotification Response Objects

```
<xs:element name = "SubscribeNotificationResponse">
     <xs:complexType>
          <xs:sequence>
               <xs:element name="TransactionStatus" type="xs:string"/>
               <xs:element name="TransactionStatusDetail"
               type="epm:TransactionStatusDetailType"/>
               <xs:element name="TransactionKey" type="epm:TransactionKeyType"
nillable="true"/>
          </xs:sequence>
     </xs:complexType>
</xs :element>
```

TransactionStatus – A string element representing the result of the overall transaction. ‘0’ denotes success. ‘1’ denotes that a warning is generated. Transaction failure is denoted by a specific numerical value.

TransactionStatusDetail – An element whose type is a complex element defined by `TransactionStatus DetailType`. See UPU S43a for details related to TransactionStatus and TransactionStatus Detail.

TransactionKey – An element whose type is a complex element defined by `TransactionKeyType` (see CEN/TS 15121-1:2011). A transaction key is returned if the call to SubscribeNotification is successful. Otherwise, the value is empty.

7.2.10 UnsubscribeNotification

7.2.10.1 General

Unsubscription is often necessary, especially in situations such as the normal shutdown of a system component. This operation cancels a previously subscribed notification. This function expects the transaction key returned by a previous call to SubscribeNotification.

7.2.10.2 UnsubscribeNotification Request Element

```
<xs:element name = "UnsubscribeNotificationRequest">
     <xs:complexType>
          <xs:sequence>
               <xs:element name="TransactionKey" type="epm:TransactionKey"/>
          </xs:sequence>
     </xs:complexType>
     </xs :element>
```

TransactionKey – An element whose type is a complex element defined by `TransactionKeyType` (see CEN/TS 15121-1:2011). It denotes the transaction key returned by a successful SubscribeNotification call.

7.2.10.3 UnsubscribeNotification Response Objects

```
<xs:element name = "UnsubscribeNotificationResponse">
     <xs:complexType>
          <xs:sequence>
               <xs:element name=”TransactionStatus” type=”xs:string”/>
               <xs:element name=”TransactionStatusDetail”
type=”epm:TransactionStatusDetailType”/>
          </xs:sequence>
     </xs:complexType>
</xs :element>
```

TransactionStatus – A string element representing the result of the overall transaction. ‘0’ denotes success. ‘1’ denotes that a warning is generated. Transaction failure is denoted by a specific numerical value.

TransactionStatusDetail – An element whose type is a complex element defined by `TransactionStatusDetailType`. See CEN/TS 15121-1:2011 for details related to TransactionStatus and TransactionStatusDetail.

7.2.11 ReceiveNotification

7.2.11.1 General

The caller of SubscribeNotification is required to implement at least one ReceiveNotification callback handler to get notification of the events subscribed to. This notification is one-way so that the handler does not need to return anything to the calling PReM System.

7.2.11.2 ReceiveNotification Input Element

```
<xs:element name = "ReceiveNotificationInput">
     <xs:complexType>
          <xs:sequence>
               <xs:element name="TransactionKey" type="epm:TransactionKey"/>
               <xs:element name="EventType" type="xsd:string"/>
               <xs:element name="EventDateTime" type="xsd:datetime"/>
               <xs:element name="EventData" type="epm:QualifiedDataType" nillabe=”true”/>
          </xs:sequence>
     </xs:complexType>
</xs :element>
```

TransactionKey – An element whose type is a complex element defined by `TransactionKeyType` (see CEN/TS 15121-1:2011). It denotes the transaction key pertaining to the event notification subscribed using SubscribeNotification.

EventType – A string element describing the type of the event which has just occurred.

EventDateTime – A datetime element which denotes the time of the event which has just occurred. It is only for reference and is not a Postmarked timestamp.

EventData – A binary element which carries any other data which can be associated with the event. It is optional and event-dependent.

8 DATA STRUCTURES AND FORMATS

8.1 Introduction

This clause explains the Message Format, Evidence Types, Evidence Format and Signature Formats being used in a PReM System.

8.2 Message format

Refer to ETSI TS 102 640-1, REM-MD Envelope Structure Implementation.

8.3 Evidence types

The evidence types and attributes indicating whether they are Mandatory/Optional/Not necessary to be sent to PReM End Users are listed below.

Table 2 — Evidence types

#	Evidence Code	Evidence Name	Evidence Description	Generator	Send to PReM End Users
01	EST-MLR-MSG-ACC/REJ-DOO	Evidence of Sent – PReM Message Acceptance/Rejection – DOO	Evidence that the PReM Message sent by mailer has been Accepted/Rejected by postal operator of origin	POO	M
02	ESG-DSP-DOO	Evidence of Sign – PReM Dispatch – DO O	Evidence that PReM Dispatch has been signed by postal operator of origin to indicate source of postal operator	POO	O
03	EFW-DSP-DOO	Evidence of Forward – DOO	Evidence that PReM Dispatch has been forwarded by postal operator of origin	POO	O
04	EFW-DSPACC/REJ-DOD	Evidence of Forward – Acceptance/Rejection – DOD	Evidence that PReM Dispatch from postal operator of origin has been Accepted/Rejected by postal operator of destination	POD	O
05	EFF-AER-DOO	Evidence of Failed Forward due to Authentication Error – DOO	Evidence that PReM Dispatch could not be forwarded from postal operator of origin to postal operator of destination due to authentication failure with postal operator of destination	POO	M

06	EFF-UNR-DOO	Evidence of Failed Forward due to Unreachable DOD – DOO	Evidence that a PReM Dispatch could not be forwarded from postal operator of origin to postal operator of destination due to unreachability of postal operator of destination	POO	M
07	ESN-MLR-MSG-ACC/ REJ-DOO	Evidence of Successful Notification – PReM Message Acceptance/ Rejection – DOO	Evidence that postal operator of origin has notified mailer that PReM Message sent by mailer had been Accepted/Rejected	POO	N
08	EFN-MLRMSG-ACC/REJDOO	Evidence of Failed Notification – PReM Message Acceptance/ Rejection – DOO	Evidence that postal operator of origin has failed to notify mailer that PReM Message sent by mailer had been Accepted/Rejected	POO	N
09	ESN-DOODSP-ACC/REJDOD	Evidence of Successful Notification – Acceptance/Rejection – DOD	Evidence that postal operator of destination has notified postal operator of origin about Acceptance/Rejection PReM Dispatch	POD	N
10	EFN-DOO-DSPACC/ REJ-DOD	Evidence of Failed Notification - Acceptance/Rejection – DOD	Evidence that postal operator of destination has failed to notify postal operator of origin about Accepted/ Rejection of PReM Dispatch	POD	N
11	ESN-MLR-TIC-DOD- SUC/ USC-DOO	Evidence of Successful Notification - Successfully/ Unsuccess fully Taken in Charge – DOD	Evidence that postal operator of origin has notified that PReM Message had been taken in charge by postal operator of destination	POO	N
12	EFN-MLR-TIC-DOD-SUC/ USC-DOO	Evidence of Failed Notification - Successfully/ Unsuccess fully Taken in Charge – DOD	Evidence that postal operator of origin has failed to notify mailer that PReM Message had been taken in charge by postal operator of destination	POO	N
13	ESN-ADR-MSG- ARR-DOD	Evidence of Successful Notification – PReM Message Arrival – DOD	Evidence that postal operator of destination has notified addressee/mailee that PReM Message addressed to addressee/mailee had arrived to postal operator of destination	POD	N
14	EFN-ADRMSG-ARR-DOD	Evidence of Failed Notification – PReM Message Arrival – DOD	Evidence that postal operator of destination has failed to notify addressee/mailee that PReM Message addressed to addressee/mailee had arrived to postal operator of destination	POD	N
15	ESN-DOOMSG-DLV/NDLDOD	Evidence of Successful Notification – Delivery/ Non-delivery – DOD	Evidence that postal operator of destination has notified postal operator of origin of PReM Message Delivery/Non-delivery	POD	N
16	EFN-DOOMSG-DLV/NDLDOD	Evidence of Failed Notification – Delivery/ Non-delivery – DOD	Evidence that postal operator of destination has failed to notify postal operator of origin of PReM Message Delivery/Non-delivery	POD	N

#	Evidence Code	Evidence Name	Evidence Description	Generator	Send to PReM End Users
17	ESN-MLRMSG-DLV/NDLDOO	Evidence of Successful Notification – Delivery/ Non-delivery – DOO	Evidence that postal operator of origin has notified mailer of PReM Message Delivery/Non-delivery	POO	N
18	EFN-MLRMSG-DLV/NDLDOO	Evidence of Failed Notification – Delivery/ Non-delivery – DOO	Evidence that postal operator of origin has failed to notify mailer of PReM Message Delivery/Non-delivery	POO	N
19	ESN-DOOMSG-EXP-DOD	Evidence of Successful Notification – PReM Message Expiration – DOD	Evidence that postal operator of destination has notified postal operator of origin that PReM Message had been expired due to non-retrieval	POD	N
20	EFN-DOO-MSG- EXP-DOD	Evidence of Failed Notification – PReM Message Expiration – DOD	Evidence that postal operator of destination has failed to notify postal operator of origin that PReM Message had been expired due to non-retrieval	POD	N
21	ESN-MLRMSG-EXP-DOO	Evidence of Successful Notification – PReM Message Expiration – DOO	Evidence that postal operator of origin has notified mailer that PReM Message had been expired due to non-retrieval	POO	N
22	EFN-MLRMSG-EXP-DOO	Evidence of Failed Notification – PReM Message Expiration – DOO	Evidence that postal operator of origin has failed to notify mailer that PReM Message had been expired due to nonretrieval	POO	N
23	ESV-DSP-DOD	Evidence of PReM Dispatch Signature Verification – DOD	Evidence that PReM Dispatch's Signature has been verified against postal operator of origin's signature public key	POD	O
24	EPV-DSP-DOD	Evidence of PReM Dispatch Postmark Verification – DOD	Evidence that the integrity of PReM Dispatch has been verified using CheckIntegrity on PReM Dispatch's Postmark	POD	O
25	EPS-DSP-DOO	Evidence of Postmark – PReM Dispatch Sent – DOO	Evidence that a Postmark operation has been invoked for the send operation of PReM Dispatch from postal operator of origin to postal operator of destination	POO	O
26	EPR-DSP-DOO	Evidence of Postmark – PReM Message Received – DOO	Evidence that a Postmark operation has been invoked for the receipt operation of PReM Message by postal operator of origin	POO	O

#	Evidence Code	Evidence Name	Evidence Description	Generator	Send to PReM End Users
27	E-DSPACC/ REJ-DOD	Evidence of PReM Dispatch Acceptance/ Rejection – DOD	Evidence that PReM Dispatch from postal operator of origin has been Accepted/Rejected by postal operator of destination	POD	O
28	E-DSP-TICSUC/ USC-DOD	Evidence of Successfully/ Unsuccessfully Taken in Charge – DOD	Evidence that PReM Message transmitted from postal operator of origin to postal operator of postal has been Successfully/Unsuccessfully Taken in Charge by postal operator of destination	POD	O
29	E-MSG-ADRDLV/ NDL-DOD	Evidence of Delivery/Non-delivery - addressee/mailee	Evidence that PReM Message has been delivered/has not been delivered by postal operator of postal to addressee/mailee	POD	M
30	E-MSG-ADREXP-DOD	Evidence of PReM Message Expiration – addressee/mailee	Evidence that PReM Message has not been retrieved from postal operator of destination by addressee/mailee within a given time period	POD	M
31	E-MSG-ADRREJ-DOD	Evidence of Reject – addressee/mailee	Evidence that PReM Message stored in postal operator of destination has been rejected by addressee/mailee	POD	M
32	EFF-MSG-ADREXP-DOO	Evidence of Failed Forward due to Expiration – DOO	Evidence that PReM Dispatch has not been forwarded from postal operator of origin to postal operator of destination within a given time period	POO	O
M = Mandatory O = Optional N = Not necessary POO = postal operator of origin POD = postal operator of destination					

8.4 Evidence format

Refer to ETSI TS 102 640-1, REM-MD Evidence Content and Semantics

8.5 Signature format

Refer to ETSI TS 102 640-1, REM Signatures

9 Policy considerations

9.1 Introduction

The PReM service between participating postal operators shall be regulated by PReM Policies defined in a PReM Policy Domain which shall reflect and complement a set of rules enforced for provision of PReM services. The PReM Policies shall specify the provisions required to provide the PReM service across borders. Participating postal operators shall, by common consent, be obliged to comply with the provisions contained in the PReM Policy Domain. Aspects which are not expressly governed by the PReM Policy Domain shall be subject to the appropriate provisions of the Acts of the Universal Postal Union (UPU).

9.2 Identity management and authentication models

The ability to generate evidence on operations and transactions is an essential feature of the PReM System. In order to provide evidence of transactions occurring in the PReM System, authentication of the mailer, addressee, postal operator of origin and postal operator of destination is necessary so that non-repudiation of the transaction by the corresponding actor could be assured.

Various identification management and authentication models, resulting in different trust levels, are possible. These include, but are not necessarily limited to:

— **Password based authentication:** Using username and password or two factor authentication mechanisms such as one-time password could be considered as the first level of identity management and authentication. Assignment of these basic authentication mechanisms should be carried out with face-to face identification, so that the postal operator can authenticate the identity of the mailer or addressee/mailee concerned.

— **Advanced Electronic Signature using electronic certificate:** This identification management and authentication model applies not only to mailer and addressee/mailee but also to postal operators. The format and rules of the certificate should conform to the standard specified in RFC 5280 and European Directive 1999/93/EC. In addition, the non-repudiation of a postal operator is handled by using the operational verb “Sign”. The Sign operation is a ‘special use’ operation normally used in a ‘Corporate Seal’ scenario when a particular subscribing organisation wants to sign something with an organisational or rolebased identity, therefore providing identity attributes and transaction evidence to the PReM Dispatch, PReM Message and Evidence. Further details of Sign are provided in CEN/TS 15121-1:2011.

— **Qualified Electronic Signature using Qualified Certificate in Secure Signature Creation Device:** This model is considered to be the highest level of identification management and authentication which a PReM postal operator can support. In accordance with RFC 3739 [2], ETSI TS 101 862 and European Directive 1999/93/EC, a subscriber of a Qualified Certificate is required to be authenticated face-to-face and the Qualified certificate profile must be used, while certificate storage media must be type approved as a Secure Signature Creation Device (defined in European Committee for Standardization, CEN/ISSS Workshop Agreement CWA 14169).

It is the decision of the postal operators as to which model or models to support in their PReM System. The level of trust between postal operators will be decided based on which of the above mechanisms has been adopted.

Annex A
(normative)

PReM XML Schema V1.00

This annex contains the full expanded content of version V1.00 of the PReM Interface Schema XML .xsd file, the version applicable as at the date of publication of this version of the standard.

```
<schema targetNamespace="http://www.upu.int/PReMService/schemas"
xmlns:prem="http://www.upu.int/PReMService/schemas"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
<import namespace="http://www.w3.org/2000/09/xmldsig#"
schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-schema.xsd"/>
     <!-- PReMService_UPU_V1.00.xsd Version 1.00 Last Updated June 15, 2008-->

     <complexType name="TransactionStatusDetailType">
          <sequence>
               <element name="ErrorNumber" type="xs:string"/>
               <element name="ErrorMessage" type="xs:string" nillable="true"/>
          </sequence>
     </complexType>
     <complexType name="TransactionKeyType">
          <sequence>
               <element name="Locator" type="prem:LocatorType"/>
               <element name="Key" type="xs:string"/>
               <element name="Sequence" type="xs:positiveInteger" nillable="true"/>
          </sequence>
     </complexType>
     <complexType name="LocatorType">
          <sequence>
               <element name="CountryCode" type="xs:string"/>
               <element name="Version" type="xs:string"/>
               <element name="ServiceProvider" type="xs:string" nillable="true"/>
               <element name="Environment" type="xs:string" nillable="true"/>
          </sequence>
     </complexType>
     <complexType name="ClaimedIdentityType">
          <sequence>
               <element name="Name" type="prem:NameIdentifierType"/>
               <element name="SupportingInfo" type="prem:SupportingInfoType"/>
          </sequence>
     </complexType>
     <complexType name="SupportingInfoType">
          <sequence>
               <element name="BasicAuth" type="prem:BasicAuthType" nillable="true"/>
               <element name="RequesterSignature" type="prem:QualifiedDataType"
nillable="true"/>
               <element name="AlternateIdentity" type="prem:AlternateIdentityType"
nillable="true"/>
          </sequence>
     </complexType>
     <complexType name="AlternateIdentityType" abstract="true">
          <sequence>
               <element name="IdentityToken" type="xs:anyType"/>
          </sequence>
     </complexType>
     <complexType name="YourFavoriteIdentityTokenType">
          <complexContent>
               <extension base="prem:AlternateIdentityType">
                    <sequence>
                         <element name="FirstElement" type="xs:string"/>
                         <element name="SecondElement" type="xs:base64Binary"/>
                    </sequence>
               </extension>
          </complexContent>
     </complexType>
     <complexType name="SignaturePolicyIdentifierType" abstract="true">
          <sequence>
               <element name="SignaturePolicyIdentifier" type="xs:anyType"/>
          </sequence>
     </complexType>
     <complexType name="SomeSignaturePolicyIdentifierType">
          <complexContent>
               <extension base="prem:SignaturePolicyIdentifierType">
                    <sequence>
                         <element name="SigPolicyID" type="xs:anyURI"/>
                         <element name="SigPolicyURL" type="xs:string"/>
                         <element name="SigPolicyHashAlgo" type="xs:anyURI"/>
                         <element name="SigPolicyHashValue" type="xs:string"/>
                         <element name="SigPolicyUserNotice" type="xs:string"
nillable="true"/>
                    </sequence>
               </extension>
          </complexContent>
     </complexType>
     <complexType name="GenericValidationDataType" abstract="true">
          <sequence>
               <element name="GenericValidationData" type="xs:anyType"/>
          </sequence>
     </complexType>
     <complexType name="X509ValidationDataType">
          <complexContent>
               <extension base="prem:GenericValidationDataType">
                    <sequence>
                         <element name="X509ValidationData"
type="prem:QualifiedDataType"/>
                    </sequence>
               </extension>
          </complexContent>
     </complexType>
     <complexType name="BasicAuthType">
          <sequence>
               <element name="UserID" type="xs:string"/>
               <element name="Password" type="xs:string" nillable="true"/>
          </sequence>
     </complexType>
     <complexType name="NameIdentifierType">
          <simpleContent>
               <extension base="xs:string">
                    <attribute name="NameQualifier" type="xs:string" use="optional"/>
                    <attribute name="Format" type="xs:anyURI" use="optional"/>
               </extension>
          </simpleContent>
     </complexType>
     <complexType name="QualifiedDataType">
          <simpleContent>
               <extension base="xs:base64Binary">
                    <attribute name="MimeType" type="xs:string" use="optional"/>
               </extension>
          </simpleContent>
     </complexType>
     <complexType name="OriginalContentType">
          <simpleContent>
               <extension base="xs:base64Binary">
                    <attribute name="MimeType" type="xs:string"/>
               </extension>
          </simpleContent>
     </complexType>
     <complexType name="ContentMetadataType">
          <sequence>
               <element name="Name" type="xs:string"/>
               <element name="Value" type="xs:string"/>
          </sequence>
     </complexType>
     <complexType name="SignatureInfoType">
          <sequence>
               <element name="SignedContent" type="prem:QualifiedDataType"
               nillable="true"/>
               <element name="ContentHash" type="xs:string" nillable="true"/>
               <element name="ContentHashAlgo" type="xs:string" nillable="true"/>
               <element name="ContentEncryptAlgo" type="xs:string" nillable="true"/>
               <element name="SigningTime" type="xs:string" nillable="true"/>
               <element name="PKCS1" type="prem:QualifiedDataType" nillable="true"/>
          </sequence>
     </complexType>
     <complexType name="X509InfoType">
          <sequence>
               <element name="X509Subject" type="xs:string"/>
               <element name="X509Issuer" type="xs:string" nillable="true"/>
               <element name="X509Serial" type="xs:string" nillable="true"/>
               <element name="X509StatusSource" type="xs:string"/>
               <element name="X509ValidFrom" type="xs:string"/>
               <element name="X509ValidTo" type="xs:string"/>
               <element name="X509Certificate" type="xs:string" nillable="true"/>
               <element name="X509RevocationReason" type="xs:string" nillable="true"/>
               <element name="X509RevocationReasonString" type="xs:string"
nillable="true"/>
               <element name="X509RevocationTime" type="xs:string" nillable="true"/>
               <element name="X509ValidationData" type="prem:X509ValidationDataType"
nillable="true"/>
          </sequence>
     </complexType>
     <complexType name="ParticipatingPartyType">
          <sequence>
               <element name="PartyName" type="prem:PartyNameType"/>
               <element name="AccessLevel" type="prem:ValidAccessLevel"/>
               <element name="NotifyEvents" type="prem:ValidOperation" nillable="true"
minOccurs="0" maxOccurs="unbounded"/>
               <element name="ContactID" type="xs:string" nillable="true"/>
          </sequence>
     </complexType>
     <complexType name="PartyNameType">
          <simpleContent>
               <extension base="xs:string">
                    <attribute name="ScopeQualifier" type="prem:Scopes" use="optional"/>
               </extension>
          </simpleContent>
     </complexType>
     <simpleType name="Scopes">
          <restriction base="xs:string">
               <enumeration value="Global"/>
               <enumeration value="Organizational"/>
               <enumeration value="Individual"/>
               <enumeration value="Mixed"/>
          </restriction>
     </simpleType>
     <simpleType name="ValidOperation">
          <restriction base="xs:string">
               <enumeration value="Verify"/>
               <enumeration value="PostMark"/>
               <enumeration value="CheckIntegrity"/>
               <enumeration value="RetrieveResults"/>
               <enumeration value="Sign"/>
               <enumeration value="StartLifecycle"/>
               <enumeration value="LogEvent"/>
```

```
            <enumeration value="Encrypt"/>
            <enumeration value="Decrypt"/>
            <enumeration value="Locate"/>
            <enumeration value="RetrieveSummary"/>
            <enumeration value="RetrievePostalAttributes"/>
      </restriction>
</simpleType>
<simpleType name="ValidOption">
      <restriction base="xs:string">
            <enumeration value="EndLifecycle"/>
            <enumeration value="ExtendLifecycle"/>
            <enumeration value="VerifyCertificate"/>
            <enumeration value="DecryptIncomingEnvelope"/>
            <enumeration value="EncryptResponse"/>
            <enumeration value="StoreNonRepudiationEvidence"/>
            <enumeration value="IssuePostMarkedReceipt"/>
            <enumeration value="ReturnTimeStampAudit"/>
            <enumeration value="ReturnSignatureInfo"/>
            <enumeration value="ReturnX509Info"/>
      </restriction>
</simpleType>
<simpleType name="ValidLocation">
      <restriction base="xs:string">
            <enumeration value="standalone"/>
            <enumeration value="embedded"/>
      </restriction>
</simpleType>
<simpleType name="ValidQualifier">
      <restriction base="xs:string">
            <enumeration value="Checked"/>
            <enumeration value="Not Checked"/>
            <enumeration value="Not Applicable"/>
      </restriction>
</simpleType>
<simpleType name="ValidAccessLevel">
      <restriction base="xs:string">
            <enumeration value="Default"/>
            <enumeration value="Signed"/>
      </restriction>
</simpleType>
<simpleType name="ValidSignatureType">
      <restriction base="xs:string">
            <enumeration value="PKCS7"/>
            <enumeration value="PKCS7-detached"/>
            <enumeration value="XMLDSIG"/>
            <enumeration value="XMLDSIG-enveloping"/>
            <enumeration value="XMLDSIG-detached"/>
            <enumeration value="XMLDSIG-template"/>
      </restriction>
</simpleType>
<simpleType name="ValidCertificateSearchType">
      <restriction base="xs:string">
            <enumeration value="Distinguished Name"/>
            <enumeration value="DN"/>
            <enumeration value="File"/>
            <enumeration value="URL"/>
      </restriction>
</simpleType>
<simpleType name="ValidImageSize">
      <restriction base="xs:string">
            <enumeration value="Small"/>
            <enumeration value="Medium"/>
            <enumeration value="Large"/>
      </restriction>
</simpleType>
<element name="PostMarkedReceipt">
      <complexType>
```

```
                <sequence>
                     <choice>
                          <element name="PKCS7SignedReceipt"
type="prem:PKCS7SignedReceiptType"/>
                          <element name="XMLSignedReceipt" type="prem:QualifiedDataType"/>
                     </choice>
                </sequence>
           </complexType>
      </element>
      <complexType name="PKCS7SignedReceiptType">
           <sequence>
                <element name="Receipt" type="prem:ReceiptType"/>
                <element name="ReceiptSignature" type="prem:QualifiedDataType"/>
           </sequence>
      </complexType>
      <complexType name="ReceiptType">
           <sequence>
                <element name="TransactionKey" type="prem:TransactionKeyType"/>
                <element name="Requester" type="xs:string" nillable="true"/>
                <element name="Operation" type="prem:ValidOperation"/>
                <element name="TSAX509SubjectName" type="xs:string"/>
                <element name="TimeStampValue" type="xs:string"/>
                <element name="RevocationStatusQualifier" type="prem:ValidQualifier"/>
                <element name="TimeStampToken" type="prem:QualifiedDataType"
nillable="true" minOccurs="0" maxOccurs="1"/>
                <element name="MessageImprint" type="xs:base64Binary" nillable="true"/>
                <element name="DigestAlgo" type="xs:string" nillable="true"/>
                <element name="DataMimeType" type="xs:string"/>
                <element name="PostMarkImage" nillable="true">
                     <complexType>
                          <simpleContent>
                               <extension base="xs:base64Binary">
                                    <attribute name="Format" type="xs:string"
default="JPG"/>
                                    <attribute name="Size" type="prem:ValidImageSize"
default="Small"/>
                               </extension>
                          </simpleContent>
                     </complexType>
                </element>
                <element name="ReceiptMetadata" type="prem:ReceiptMetadataType"
nillable="true" minOccurs="0" maxOccurs="unbounded"/>
           </sequence>
      </complexType>
      <complexType name="ReceiptMetadataType">
           <sequence>
                <element name="Name" type="xs:string"/>
                <choice>
                     <element name="Value" type="xs:string"/>
                     <element name="EncodedValue" type="prem:QualifiedDataType"/>
                </choice>
           </sequence>
      </complexType>
      <complexType name="ClientApplicationType">
           <sequence>
                <element name="NameAndVersion" type="xs:string"/>
                <element name="ContentTransformScheme"
                type="prem:ContentTransformSchemeType"
nillable="true"/>
           </sequence>
      </complexType>
      <complexType name="ContentTransformSchemeType">
           <simpleContent>
                <extension base="xs:string">
                     <attribute name="ContentTransformSchemeURI" type="xs:anyURI"
use="optional"/>
                </extension>
```

```
          </simpleContent>
     </complexType>
     <complexType name="IssuePostMarkedReceiptType">
          <sequence>
               <element name="Location" type="prem:ValidLocation" minoccurs="0"/>
               <element name="PostMarkImage" type="prem:PostMarkImageType"
minoccurs="0"/>
          </sequence>
     </complexType>
     <complexType name="PostMarkImageType">
          <simpleContent>
               <extension base="xs:boolean">
                    <attribute name="Format" type="xs:string" default="JPG"/>
                    <attribute name="Size" type="prem:ValidImageSize" default="Small"/>
               </extension>
          </simpleContent>
     </complexType>
     <complexType name="SignatureSelectorType">
          <sequence>
               <choice>
                    <element name="NodeName" type="xs:string" maxOccurs="unbounded"/>
                    <element name="XPathSelector" type="prem:XPathSelectorType"/>
               </choice>
          </sequence>
     </complexType>
     <complexType name="XPathSelectorType">
          <sequence>
               <element name="XPath" type="xs:string"/>
               <element name="NameSpace" type="xs:string" nillable="true"/>
               <element name="Qualifer" type="xs:string" nillable="true"/>
          </sequence>
     </complexType>
     <element name="VerifyRequest">
          <complexType>
               <sequence>
                    <element name="VerifyOptions" type="prem:VerifyOptionsType"/>
                    <element name="Version" type="xs:string"/>
                    <element name="TransactionKey" type="prem:TransactionKeyType"
nillable="true"/>
                    <element name="ClaimedIdentity" type="prem:ClaimedIdentityType"
nillable="true"/>
                    <element name="OrganizationID" type="xs:string" nillable="true"/>
                    <element name="ClientApplication" type="prem:ClientApplicationType"/>
                    <element name="ContentIdentifier" type="xs:string" nillable="true"/>
                    <element name="SignatureData" type="prem:QualifiedDataType"
nillable="true"/>
                    <element name="SignedContent" type="prem:QualifiedDataType"
nillable="true"/>
                    <element ref="prem:PostMarkedReceipt" minOccurs="0"/>
                    <element name="SignatureSelector" type="prem:SignatureSelectorType"
nillable="true"/>
                    <element name="ContentMetadata" type="prem:ContentMetadataType"
nillable="true" minOccurs="0" maxOccurs="unbounded"/>
               </sequence>
          </complexType>
    </element>
     <complexType name="VerifyOptionsType">
          <sequence>
               <element name="EndLifecycle" type="xs:boolean"/>
               <element name="ExtendLifecycle" type="xs:boolean"/>
               <element name="VerifyCertificate" type="xs:boolean"/>
               <element name="DecryptIncomingEnvelope" type="xs:boolean"/>
               <element name="EncryptResponse" type="xs:boolean"/>
               <element name="StoreNonRepudiationEvidence" type="xs:boolean"/>
               <element name="IssuePostMarkedReceipt"
               type="prem:IssuePostMarkedReceiptType"
               nillable="true"/>
```

```
                <element name="ReturnSignatureInfo" type="xs:boolean"/>
                <element name="ReturnX509Info" type="xs:boolean"/>
            </sequence>
        </complexType>
        <element name="VerifyResponse">
            <complexType>
                <sequence>
                    <element name="TransactionStatus" type="xs:string"/>
                    <element name="TransactionStatusDetail"
type="prem:TransactionStatusDetailType" nillable="true" minOccurs="0"
maxOccurs="unbounded"/>
                    <element name="TransactionKey" type="prem:TransactionKeyType"/>
                    <element ref="prem:PostMarkedReceipt" minOccurs="0"/>
                    <element name="SignatureData" type="prem:QualifiedDataType"
nillable="true"/>
                    <element name="SignatureInfo" type="prem:SignatureInfoType"
nillable="true"/>
                    <element name="X509Info" type="prem:X509InfoType" nillable="true"/>
                </sequence>
            </complexType>
        </element>
        <element name="PostMarkRequest">
            <complexType>
                <sequence>
                    <element name="PostMarkOptions" type="prem:PostMarkOptionsType"/>
                    <element name="Version" type="xs:string"/>
                    <element name="SignatureType" type="prem:ValidSignatureType"/>
                    <element name="TransactionKey" type="prem:TransactionKeyType"
nillable="true"/>
                    <element name="ClaimedIdentity" type="prem:ClaimedIdentityType"
nillable="true"/>
                    <element name="OrganizationID" type="xs:string" nillable="true"/>
                    <element name="ClientApplication" type="prem:ClientApplicationType"/>
                    <element name="ContentIdentifier" type="xs:string" nillable="true"/>
                    <element name="Data" type="prem:QualifiedDataType"/>
                    <xs:element name="PostMarkImage" type="prem:PostMarkImageType"
nillable="true" minOccurs="0"/>
                    <element name="ContentMetadata" type="prem:ContentMetadataType"
nillable="true" minOccurs="0" maxOccurs="unbounded"/>
                </sequence>
            </complexType>
        </element>
        <complexType name="PostMarkOptionsType">
            <sequence>
                <element name="EndLifecycle" type="xs:boolean"/>
                <element name="ExtendLifecycle" type="xs:boolean"/>
                <element name="DecryptIncomingEnvelope" type="xs:boolean"/>
                <element name="EncryptResponse" type="xs:boolean"/>
                <element name="StoreNonRepudiationEvidence" type="xs:boolean"/>
            </sequence>
        </complexType>
        <element name="PostMarkResponse">
            <complexType>
                <sequence>
                    <element name="TransactionStatus" type="xs:string"/>
                    <element name="TransactionStatusDetail"
type="prem:TransactionStatusDetailType" nillable="true" minOccurs="0"
maxOccurs="unbounded"/>
                    <element name="TransactionKey" type="prem:TransactionKeyType"/>
                    <element ref="prem:PostMarkedReceipt"/>
                    <element name="SignatureData" type="prem:QualifiedDataType"
nillable="true"/>
                </sequence>
            </complexType>
        </element>
        <element name="CheckIntegrityRequest">
            <complexType>
```

```
                <sequence>
                     <element name="CheckIntegrityOptions"
type="prem:CheckIntegrityOptionsType"/>
                     <element name="Version" type="xs:string"/>
                     <element name="SignatureType" type="prem:ValidSignatureType"
nillable="true"/>
                     <element name="TransactionKey" type="prem:TransactionKeyType"/>
                     <element name="ClaimedIdentity" type="prem:ClaimedIdentityType"
nillable="true"/>
                     <element name="OrganizationID" type="xs:string" nillable="true"/>
                     <element name="ClientApplication" type="prem:ClientApplicationType"/>
                     <element name="ContentIdentifier" type="xs:string" nillable="true"/>
                     <element name="OriginalContent" type="prem:OriginalContentType"
maxOccurs="unbounded"/>
                </sequence>
           </complexType>
      </element>
      <complexType name="CheckIntegrityOptionsType">
           <sequence>
                <element name="DecryptIncomingEnvelope" type="xs:boolean"/>
                <element name="StoreNonRepudiationEvidence" type="xs:boolean"/>
                <element name="IssuePostMarkedReceipt"
type="prem:IssuePostMarkedReceiptType" nillable="true"/>
           </sequence>
      </complexType>
      <element name="CheckIntegrityResponse">
           <complexType>
                <sequence>
                     <element name="TransactionStatus" type="xs:string"/>
                     <element name="TransactionStatusDetail"
type="prem:TransactionStatusDetailType" nillable="true" minOccurs="0"
maxOccurs="unbounded"/>
                     <element name="TransactionKey" type="prem:TransactionKeyType"/>
                     <element ref="prem:PostMarkedReceipt" minOccurs="0"/>
                </sequence>
           </complexType>
      </element>
      <element name="LogEventRequest">
           <complexType>
                <sequence>
                     <element name="LogEventOptions" type="prem:LogEventOptionsType"/>
                     <element name="Version" type="xs:string"/>
                     <element name="TransactionKey" type="prem:TransactionKeyType"
nillable="true"/>
                     <element name="ClaimedIdentity" type="prem:ClaimedIdentityType"
nillable="true"/>
                     <element name="OrganizationID" type="xs:string" nillable="true"/>
                     <element name="ClientApplication" type="prem:ClientApplicationType"/>
                     <element name="ContentIdentifier" type="xs:string" nillable="true"/>
                     <element name="Data" type="prem:QualifiedDataType"/>
                     <element name="ContentMetadata" type="prem:ContentMetadataType"
nillable="true" minOccurs="0" maxOccurs="unbounded"/>
                </sequence>
           </complexType>
      </element>
      <complexType name="LogEventOptionsType">
           <sequence>
                <element name="EndLifecycle" type="xs:boolean"/>
                <element name="ExtendLifecycle" type="xs:boolean"/>
                <element name="DecryptIncomingEnvelope" type="xs:boolean"/>
           </sequence>
      </complexType>
      <element name="LogEventResponse">
           <complexType>
                <sequence>
                     <element name="TransactionStatus" type="xs:string"/>
                     <element name="TransactionStatusDetail"
```

```
type="prem:TransactionStatusDetailType" nillable="true" minOccurs="0"
maxOccurs="unbounded"/>
                    <element name="TransactionKey" type="prem:TransactionKeyType"/>
               </sequence>
          </complexType>
     </element>
     <element name="RetrieveResultsRequest">
          <complexType>
               <sequence>
                    <element name="RetrieveResultsOptions"
type="prem:RetrieveResultsOptionsType"/>
                    <element name="Version" type="xs:string"/>
                    <element name="SignatureType" type="prem:ValidSignatureType"
nillable="true"/>
                    <element name="TransactionKey" type="prem:TransactionKeyType"/>
                    <element name="ClaimedIdentity" type="prem:ClaimedIdentityType"
nillable="true"/>
                    <element name="OrganizationID" type="xs:string" nillable="true"/>
                    <element name="ClientApplication" type="prem:ClientApplicationType"/>
                    <element name="ContentIdentifier" type="xs:string" nillable="true"/>
               </sequence>
          </complexType>
     </element>
     <complexType name="RetrieveResultsOptionsType">
          <sequence>
               <element name="StoreNonRepudiationEvidence" type="xs:boolean"/>
               <element name="IssuePostMarkedReceipt"
type="prem:IssuePostMarkedReceiptType" nillable="true"/>
               <element name="EncryptResponse" type="xs:boolean"/>
               <element name="ReturnTimeStampAudit" type="xs:boolean"/>
               <element name="ReturnSignatureInfo" type="xs:boolean"/>
               <element name="ReturnX509Info" type="xs:boolean"/>
          </sequence>
     </complexType>
     <element name="RetrieveResultsResponse">
          <complexType>
               <sequence>
                    <element name="TransactionStatus" type="xs:string"/>
                    <element name="TransactionStatusDetail"
type="prem:TransactionStatusDetailType" nillable="true" minOccurs="0"
maxOccurs="unbounded"/>
                    <element name="TransactionKey" type="prem:TransactionKeyType"/>
                    <element ref="prem:PostMarkedReceipt" minOccurs="0"/>
                    <element name="Results" type="prem:ResultsType"/>
               </sequence>
          </complexType>
     </element>
     <complexType name="ResultsType">
          <sequence>
               <element name="TransactionStatus" type="xs:string"/>
               <element name="TransactionStatusDetail"
type="prem:TransactionStatusDetailType" nillable="true" minOccurs="0"
maxOccurs="unbounded"/>
               <element name="TransactionKey" type="prem:TransactionKeyType"/>
               <element name="Operation" type="xs:string"/>
               <element name="OperationOptions" type="prem:ValidOption" nillable="true"
minOccurs="0" maxOccurs="unbounded"/>
               <element name="OrganizationID" type="xs:string" nillable="true"/>
               <element name="UniqueSequenceId" type="xs:string" nillable="true"/>
               <element name="ClientApplication" type="prem:ClientApplicationType"
nillable="true"/>
               <element name="ContentIdentifier" type="xs:string" nillable="true"/>
               <element ref="prem:PostMarkedReceipt" minOccurs="0"/>
               <element name="Data" type="prem:QualifiedDataType" nillable="true"
minOccurs="0" maxOccurs="unbounded"/>
               <element name="ContentMetadata" type="prem:ContentMetadataType"
nillable="true" minOccurs="0" maxOccurs="unbounded"/>
```

```
                <element name="TimeStampAudit" type="prem:QualifiedDataType"
nillable="true" minOccurs="0" maxOccurs="unbounded"/>
                <element name="SignatureInfo" type="prem:SignatureInfoType"
nillable="true"/>
                <element name="X509Info" type="prem:X509InfoType" nillable="true"/>
           </sequence>
      </complexType>
      <element name="SignRequest">
           <complexType>
                <sequence>
                     <element name="SignOptions" type="prem:SignOptionsType"/>
                     <element name="Version" type="xs:string"/>
                     <element name="TransactionKey" type="prem:TransactionKeyType"
nillable="true"/>
                     <element name="ClaimedIdentity" type="prem:ClaimedIdentityType"
nillable="true"/>
                     <element name="OrganizationID" type="xs:string" nillable="true"/>
                     <element name="ClientApplication" type="prem:ClientApplicationType"/>
                     <element name="ContentIdentifier" type="xs:string" nillable="true"/>
                     <element name="Data" type="prem:QualifiedDataType"/>
                     <element name="SignatureType" type="prem:ValidSignatureType"/>
                     <element name="KeyName" type="xs:string" nillable="true"/>
                     <element name="SignaturePolicyID" type="xs:anyURI" nillable="true"/>
                     <element name="ContentMetadata" type="prem:ContentMetadataType"
nillable="true" minOccurs="0" maxOccurs="unbounded"/>
                </sequence>
           </complexType>
      </element>
      <complexType name="SignOptionsType">
           <sequence>
                <element name="EndLifecycle" type="xs:boolean"/>
                <element name="ExtendLifecycle" type="xs:boolean"/>
                <element name="VerifyCertificate" type="xs:boolean"/>
                <element name="IssuePostMarkedReceipt"
                type="prem:IssuePostMarkedReceiptType"
                nillable="true"/>
                <element name="StoreNonRepudiationEvidence" type="xs:boolean"/>
                <element name="DecryptIncomingEnvelope" type="xs:boolean"/>
                <element name="EncryptResponse" type="xs:boolean"/>
                <element name="ReturnSignatureInfo" type="xs:boolean"/>
                <element name="ReturnX509Info" type="xs:boolean"/>
           </sequence>
      </complexType>
      <element name="SignResponse">
           <complexType>
                <sequence>
                     <element name="TransactionStatus" type="xs:string"/>
                     <element name="TransactionStatusDetail"
type="prem:TransactionStatusDetailType" nillable="true" minOccurs="0"
maxOccurs="unbounded"/>
                     <element name="TransactionKey" type="prem:TransactionKeyType"/>
                     <element ref="prem:PostMarkedReceipt" minOccurs="0"/>
                     <element name="SignatureData" type="prem:QualifiedDataType"
nillable="true"/>
                     <element name="SignatureInfo" type="prem:SignatureInfoType"
nillable="true"/>
                     <element name="X509Info" type="prem:X509InfoType" nillable="true"/>
                </sequence>
           </complexType>
      </element>
      <element name="EncryptRequest">
           <complexType>
                <sequence>
                     <element name="EncryptOptions" type="prem:EncryptOptionsType"/>
                     <element name="Version" type="xs:string"/>
                     <element name="TransactionKey" type="prem:TransactionKeyType"
nillable="true"/>
```

```
                    <element name="ClaimedIdentity" type="prem:ClaimedIdentityType"
nillable="true"/>
                    <element name="OrganizationID" type="xs:string" nillable="true"/>
                    <element name="ClientApplication" type="prem:ClientApplicationType"/>
                    <element name="ContentIdentifier" type="xs:string" nillable="true"/>
                    <element name="Data" type="prem:QualifiedDataType"/>
                    <element name="SignatureType" type="prem:ValidSignatureType"/>
                    <element name="NodeName" type="xs:string" nillable="true"/>
                    <element name="SessionKeyAlgo" type="xs:string" nillable="true"/>
                    <element name="CertificateSearchType"
type="prem:ValidCertificateSearchType"/>
                    <element name="CertificateID" type="xs:string" minOccurs="1"
maxOccurs="unbounded"/>
                    <element name="ContentMetadata" type="prem:ContentMetadataType"
nillable="true" minOccurs="0" maxOccurs="unbounded"/>
               </sequence>
          </complexType>
     </element>
     <complexType name="EncryptOptionsType">
          <sequence>
               <element name="EndLifecycle" type="xs:boolean"/>
               <element name="ExtendLifecycle" type="xs:boolean"/>
               <element name="VerifyCertificate" type="xs:boolean"/>
               <element name="StoreNonRepudiationEvidence" type="xs:boolean"/>
               <element name="IssuePostMarkedReceipt"
type="prem:IssuePostMarkedReceiptType" nillable="true"/>
               <element name="DecryptIncomingEnvelope" type="xs:boolean"/>
               <element name="ReturnSignatureInfo" type="xs:boolean"/>
               <element name="ReturnX509Info" type="xs:boolean"/>
          </sequence>
     </complexType>
     <element name="EncryptResponse">
     <complexType>
          <sequence>
               <element name="TransactionStatus" type="xs:string"/>
               <element name="TransactionStatusDetail"
type="prem:TransactionStatusDetailType" nillable="true" minOccurs="0"
maxOccurs="unbounded"/>
               <element name="TransactionKey" type="prem:TransactionKeyType"/>
               <element ref="prem:PostMarkedReceipt" minOccurs="0"/>
               <element name="SignatureData" type="prem:QualifiedDataType"/>
               <element name="SignatureInfo" type="prem:SignatureInfoType"
nillable="true"/>
               <element name="X509Info" type="prem:X509InfoType" nillable="true"/>
          </sequence>
     </complexType>
     </element>
     <element name="DecryptRequest">
          <complexType>
               <sequence>
                    <element name="DecryptOptions" type="prem:DecryptOptionsType"/>
                    <element name="Version" type="xs:string"/>
                    <element name="TransactionKey" type="prem:TransactionKeyType"
nillable="true"/>
                    <element name="ClaimedIdentity" type="prem:ClaimedIdentityType"
nillable="true"/>
                    <element name="OrganizationID" type="xs:string" nillable="true"/>
                    <element name="ClientApplication" type="prem:ClientApplicationType"/>
                    <element name="ContentIdentifier" type="xs:string" nillable="true"/>
                    <element name="EnvelopedData" type="prem:QualifiedDataType"/>
                    <element name="ContentMetadata" type="prem:ContentMetadataType"
nillable="true" minOccurs="0" maxOccurs="unbounded"/>
               </sequence>
          </complexType>
     </element>
     <complexType name="DecryptOptionsType">
          <sequence>
```

```
                <element name="EndLifecycle" type="xs:boolean"/>
                <element name="ExtendLifecycle" type="xs:boolean"/>
                <element name="StoreNonRepudiationEvidence" type="xs:boolean"/>
                <element name="EncryptResponse" type="xs:boolean"/>
                <element name="ReturnSignatureInfo" type="xs:boolean"/>
                <element name="ReturnX509Info" type="xs:boolean"/>
            </sequence>
        </complexType>
        <element name="DecryptResponse">
            <complexType>
                <sequence>
                    <element name="TransactionStatus" type="xs:string"/>
                    <element name="TransactionStatusDetail"
type="prem:TransactionStatusDetailType" nillable="true" minOccurs="0"
maxOccurs="unbounded"/>
                    <element name="TransactionKey" type="prem:TransactionKeyType"/>
                    <element name="Data" type="prem:QualifiedDataType"/>
                    <element name="SignatureInfo" type="prem:SignatureInfoType"
nillable="true"/>
                    <element name="X509Info" type="prem:X509InfoType" nillable="true"/>
                </sequence>
            </complexType>
        </element>
        <element name="LocateRequest">
            <complexType>
                <sequence>
                    <element name="LocateOptions" type="prem:LocateOptionsType"/>
                    <element name="Version" type="xs:string"/>
                    <element name="TransactionKey" type="prem:TransactionKeyType"
nillable="true"/>
                    <element name="ClaimedIdentity" type="prem:ClaimedIdentityType"
nillable="true"/>
                    <element name="OrganizationID" type="xs:string" nillable="true"/>
                    <element name="ClientApplication" type="prem:ClientApplicationType"/>
                    <element name="CertificateSearchType"
type="prem:ValidCertificateSearchType"/>
                    <element name="CertificateID" type="xs:string"/>
                </sequence>
            </complexType>
        </element>
        <complexType name="LocateOptionsType">
            <sequence>
                <element name="EndLifecycle" type="xs:boolean"/>
                <element name="ExtendLifecycle" type="xs:boolean"/>
                <element name="VerifyCertificate" type="xs:boolean"/>
                <element name="ReturnX509Info" type="xs:boolean"/>
            </sequence>
        </complexType>
        <element name="LocateResponse">
            <complexType>
                <sequence>
                    <element name="TransactionStatus" type="xs:string"/>
                    <element name="TransactionStatusDetail"
type="prem:TransactionStatusDetailType" nillable="true" minOccurs="0"
maxOccurs="unbounded"/>
                    <element name="TransactionKey" type="prem:TransactionKeyType"/>
                    <element name="PublicEncryptionCert" type="xs:string"/>
                    <element name="X509Info" type="prem:X509InfoType" nillable="true"/>
                </sequence>
            </complexType>
        </element>
        <element name="RetrievePostalAttributesRequest">
            <complexType>
                <sequence>
                    <element name="Locator" type="prem:LocatorType"/>
                    <element name="LanguageCode" type="xs:string"/>
                    <element name="ClientApplication" type="prem:ClientApplicationType"
```

```
nillable="true"/>
                    <element name="AttributeCategory" type="xs:string" nillable="true"/>
                </sequence>
            </complexType>
        </element>
        <complexType name="PostalAttributeType">
            <sequence>
                <element name="AttributeName" type="xs:string"/>
                <element name="LastModifiedDateTime" type="xs:string"/>
                <element name="ExpiresOnDateTime" type="xs:string"/>
                <element name="AttributeTextValue" type="xs:string" nillable="true"/>
                <element name="AttributeBinaryValue" type="xs:base64Binary"
nillable="true"/>
            </sequence>
        </complexType>
        <element name="RetrievePostalAttributesResponse">
            <complexType>
                <sequence>
                    <element name="TransactionStatus" type="xs:string"/>
                    <element name="TransactionStatusDetail"
type="prem:TransactionStatusDetailType" nillable="true" minOccurs="0"
maxOccurs="unbounded"/>
                    <element name="TransactionKey" type="prem:TransactionKeyType"/>
                    <element name="PostalAttribute" type="prem:PostalAttributeType"
nillable="true" minOccurs="0" maxOccurs="unbounded"/>
                </sequence>
            </complexType>
        </element>

<complexType name=”SendMessageToDestinationOptionsType”>
     <sequence>
          <element name=”EndLifeCycle” type=”xs:boolean”/>
          <element name=”ExtendLifeCycle” type=”xs:boolean”/>
          <element name=”IssuePostMarkedReceipt” type=”xs:boolean” />
     </sequence>
</complexType>

< element name=”SendMessageToDestinationRequest”>
     < complexType>
          < sequence>
               < element name=”SendMessageToDestinationOptions”
type=”prem:SendMessageToDestinationOptionsType”/>
               < element name=”TransactionKey” type=”epm:TransactionKeyType” />
          < element name=”OriginalClaimedIdentity” type=”epm:ClaimedIdentityType” />
               < element name=”ClaimedIdentity” type=”epm:ClaimedIdentityType” />
               < element name=”OrganizationID” type=”xs:string” nillable=”true”/>
               < element name=”ClientApplication”type=”epm:ClientApplicationType” />
               < element name=”ContentIdentifier” type=”xs:string” nillable=”true” />
               < element name=”Destination” type=”xs:string” />
               < element name=”Data” type=”epm:QualifiedDataType” />
               < element name=”ContentMetadata” type=”epm:ContentMetadataType”
nillable=”true” minOccurs=”0” maxOccurs=”unbounded” />
          </ sequence>
     </ complexType>
</ element>

< element name=”SendMessageToDestinationResponse”>
     < complexType>
          < element name=”TransactionStatus” type=”xs:string”/>
          < element name=”TransactionStatusDetail”
type=”epm:TransactionStatusDetailType”/>
          < element name=”TransactionKey” type=”epm:TransactionKeyType”/>
          < element name=”PostMarkedReceipt” type=”epm:PostMarkedReceiptType”
          nillable=”true”/>
     </ complexType>
</ element>
```

```
< element name=”RejectMessageRequest”>
     < complexType>
          < sequence>
               < element name=”TransactionKey” type=”epm:TransactionKeyType” />
          </ sequence>
     </ complexType>
</ element>

< element name=”SendMessageToDestinationResponse”>
     < complexType>
          < element name=”TransactionStatus” type=”xs:string”/>
          < element name=”TransactionStatusDetail”
type=”epm:TransactionStatusDetailType”/>
          < element name=”TransactionKey” type=”epm:TransactionKeyType”/>
          < element name=”PostMarkedReceipt” type=”epm:PostMarkedReceiptType”
nillable=”true”/>
     </ complexType>
</ element>

< element name = "SubscribeNotificationRequest">
     < complexType>
          < sequence>
               < element name="EventType" type="xsd:string"/>
               < element name=”ClientApplication” type=”epm:ClientApplicationType” />
               < element name="CallbackUrl" type="xsd:string"/>
          </ sequence>
     </ complexType>
</ element>

< element name = "SubscribeNotificationResponse">
     < complexType>
          < sequence>
               < element name=”TransactionStatus” type=”xs:string”/>
               < element name=”TransactionStatusDetail”
type=”epm:TransactionStatusDetailType”/>
               < element name=”TransactionKey” type=”epm:TransactionKeyType”
nillable=”true”/>
          </ sequence>
     </ complexType>
</ element>

< element name = "UnsubscribeNotificationRequest">
     < complexType>
          < sequence>
               < element name="TransactionKey" type="epm:TransactionKey"/>
          </ sequence>
     </ complexType>
</ element>

< element name = "UnsubscribeNotificationResponse">
     < complexType>
          < sequence>
               < element name=”TransactionStatus” type=”xs:string”/>
               < element name=”TransactionStatusDetail”
type=”epm:TransactionStatusDetailType”/>
          </ sequence>
     </ complexType>
</ element>

< element name = "ReceiveNotificationInput">
     < complexType>
          < sequence>
               < element name="TransactionKey" type="epm:TransactionKey"/>
               < element name="EventType" type="xsd:string"/>
               < element name="EventDateTime" type="xsd:datetime"/>
               < element name="EventData" type="epm:QualifiedDataType" nillabe=”true”/>
          </ sequence>
          </ complexType>
     </ element>
</schema>
```

Annex B
(normative)

PReM Web Services Description Language (WSDL) V1.0

This annex contains the full expanded content of version V1.00 of the PReM Web Services Description Language (.wsdl) file applicable as at the date of publication of this version of this Technical Specification.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="PReMService"
           targetNamespace="http://www.upu.int/PReMService/definitions"
           xmlns="http://schemas.xmlsoap.org/wsdl/"
           xmlns:tns="http://www.upu.int/PReMService/definitions"
           xmlns:prem="http://www.upu.int/PReMService/schemas"
           xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
           xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
           xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
           xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
<wsdl:types>

<xs:schema
    targetNamespace="http://www.upu.int/PReMService/schemas"
    xmlns="http://www.w3.org/2001/XMLSchema"
    xmlns:prem="http://http://www.upu.int/PReMService/schemas">
    <xs:include schemaLocation="file://C:/PReM-UPU-WSDLs/PReMService-UPU-V1.00.xsd"/>

<!-- schemaLocation may be changed to reflect local needs as required.
Schema file may be placed on a Web Server and the above include can be changed to
http:// syntax as required. Also note that Visual Studio expects a conventional
directory path and file name in schemaLocation.
-->

</xs:schema>

</wsdl:types>

<!-- Request Response Message Definition -->

    <wsdl:message name="VerifyRequestMessage">
        <wsdl:part name="VerifyReq" element="prem:VerifyRequest"/>
    </wsdl:message>

    <wsdl:message name="VerifyResponseMessage">
        <wsdl:part name="VerifyResp" element="prem:VerifyResponse"/>
    </wsdl:message>

<!-- Request Response Message Definition -->

    <wsdl:message name="EncryptRequestMessage">
        <wsdl:part name="EncryptReq" element="prem:EncryptRequest"/>
    </wsdl:message>

    <wsdl:message name="EncryptResponseMessage">
        <wsdl:part name="EncryptResp" element="prem:EncryptResponse"/>
    </wsdl:message>

<!-- Request Response Message Definition -->

    <wsdl:message name="DecryptRequestMessage">
        <wsdl:part name="DecryptReq" element="prem:DecryptRequest"/>
    </wsdl:message>
    <wsdl:message name="DecryptResponseMessage">
        <wsdl:part name="DecryptResp" element="prem:DecryptResponse"/>
    </wsdl:message>

<!-- Request Response Message Definition -->

    <wsdl:message name="CheckIntegrityRequestMessage">
        <wsdl:part name="CheckIntegrityReq" element="prem:CheckIntegrityRequest"/>
    </wsdl:message>

    <wsdl:message name="CheckIntegrityResponseMessage">
        <wsdl:part name="CheckIntegrityResp" element="prem:CheckIntegrityResponse"/>
    </wsdl:message>

<!-- Request Response Message Definition -->

    <wsdl:message name="RetrieveResultsRequestMessage">
        <wsdl:part name="RetrieveResultsReq" element="prem:RetrieveResultsRequest"/>
    </wsdl:message>

    <wsdl:message name="RetrieveResultsResponseMessage">
        <wsdl:part name="RetrieveResultsResp" element="prem:RetrieveResultsResponse"/>
    </wsdl:message>

<!-- Request Response Message Definition -->

    <wsdl:message name="LocateRequestMessage">
        <wsdl:part name="LocateReq" element="prem:LocateRequest"/>
    </wsdl:message>

    <wsdl:message name="LocateResponseMessage">
        <wsdl:part name="LocateResp" element="prem:LocateResponse"/>
    </wsdl:message>

<!-- Request Response Message Definition -->

    <wsdl:message name="PostMarkRequestMessage">
        <wsdl:part name="PostMarkReq" element="prem:PostMarkRequest"/>
    </wsdl:message>

    <wsdl:message name="PostMarkResponseMessage">
        <wsdl:part name="PostMarkResp" element="prem:PostMarkResponse"/>
    </wsdl:message>

<!-- Request Response Message Definition -->

    <wsdl:message name="SignRequestMessage">
        <wsdl:part name="SignReq" element="prem:SignRequest"/>
    </wsdl:message>

    <wsdl:message name="SignResponseMessage">
        <wsdl:part name="SignResp" element="prem:SignResponse"/>
    </wsdl:message>

<!-- Request Response Message Definition -->

    <wsdl:message name="LogEventRequestMessage">
        <wsdl:part name="LogEventReq" element="prem:LogEventRequest"/>
    </wsdl:message>

    <wsdl:message name="LogEventResponseMessage">
        <wsdl:part name="LogEventResp" element="prem:LogEventResponse"/>
    </wsdl:message>

<!-- Request Response Message Definition -->
```

```
    <wsdl:message name="RetrievePostalAttributesRequestMessage">
        <wsdl:part name="RetrievePostalAttributesReq"
    element="prem:RetrievePostalAttributesRequest"/>
    </wsdl:message>

    <wsdl:message name="RetrievePostalAttributesResponseMessage">
        <wsdl:part name="RetrievePostalAttributesResp"
    element="prem:RetrievePostalAttributesResponse"/>
    </wsdl:message>

<!-- Request Response Message Definition -->

    <wsdl:message name="SendMessageToDestinationRequestMessag">
        <wsdl:part name=" SendMessageToDestinationReq" element="prem:
    SendMessageToDestinationRequest"/>
    </wsdl:message>

    <wsdl:message name="SendMessageToDestinationResponseMessage">
        <wsdl:part name="SendMessageToDestinationResp" element="prem:
    SendMessageToDestinationResponse"/>
    </wsdl:message>

<!-- Request Response Message Definition -->

    <wsdl:message name="SubscribeNotificationRequestMessage">
        <wsdl:part name="SubscribeNotificationReq" element="prem:
    SubscribeNotificationRequest"/>
    </wsdl:message>

    <wsdl:message name="SubscribeNotificationResponseMessage">
        <wsdl:part name="SubscribeNotificationResp" element="prem:
    SubscribeNotificationResponse"/>
    </wsdl:message>

<!-- Request Response Message Definition -->

    <wsdl:message name="UnSubscribeNotificationRequestMessage">
        <wsdl:part name="UnSubscribeNotificationReq" element="prem:
    UnSubscribeNotificationRequest"/>
    </wsdl:message>

    <wsdl:message name="UnSubscribeNotificationResponseMessage">
        <wsdl:part name="UnSubscribeNotificationResp" element="prem:
    UnSubscribeNotificationResponse"/>
    </wsdl:message>

<!-- Request Response Message Definition -->

    <wsdl:message name="ReceiveNotificationRequestMessage">
        <wsdl:part name="ReceiveNotificationReq" element="prem:
    ReceiveNotificationRequest"/>
    </wsdl:message>

    <wsdl:message name="ReceiveNotificationResponseMessage">
        <wsdl:part name="ReceiveNotificationResp" element="prem:
    ReceiveNotificationResponse"/>
    </wsdl:message>

<!-- Request Response Message Definition -->

    <wsdl:message name="RejectMessageRequestMessage">
        <wsdl:part name="RejectMessageReq" element="prem: RejectMessageRequest"/>
    </wsdl:message>

    <wsdl:message name="RejectMessageResponseMessage">
        <wsdl:part name="RejectMessageResp" element="prem: RejectMessageResponse"/>
    </wsdl:message>
```

```
<!-- Request Response Message Definition -->
<!-- WSDL schema PortType and Operation to Message mapping elements -->

<wsdl:portType name="PReMServicePortType">

    <wsdl:operation name="Verify">
        <input message="tns:VerifyRequestMessage"/>
        <output message="tns:VerifyResponseMessage"/>
    </wsdl:operation>

    <wsdl:operation name="Encrypt">
        <input message="tns:EncryptRequestMessage"/>
        <output message="tns:EncryptResponseMessage"/>
    </wsdl:operation>

    <wsdl:operation name="Decrypt">
        <input message="tns:DecryptRequestMessage"/>
        <output message="tns:DecryptResponseMessage"/>
    </wsdl:operation>

    <wsdl:operation name="CheckIntegrity">
        <input message="tns:CheckIntegrityRequestMessage"/>
        <output message="tns:CheckIntegrityResponseMessage"/>
    </wsdl:operation>

    <wsdl:operation name="RetrieveResults">
        <input message="tns:RetrieveResultsRequestMessage"/>
        <output message="tns:RetrieveResultsResponseMessage"/>
    </wsdl:operation>

    <wsdl:operation name="Locate">
        <input message="tns:LocateRequestMessage"/>
        <output message="tns:LocateResponseMessage"/>
    </wsdl:operation>

    <wsdl:operation name="PostMark">
        <input message="tns:PostMarkRequestMessage"/>
        <output message="tns:PostMarkResponseMessage"/>
    </wsdl:operation>

    <wsdl:operation name="Sign">
        <input message="tns:SignRequestMessage"/>
        <output message="tns:SignResponseMessage"/>
    </wsdl:operation>

    <wsdl:operation name="LogEvent">
        <input message="tns:LogEventRequestMessage"/>
        <output message="tns:LogEventResponseMessage"/>
    </wsdl:operation>

    <wsdl:operation name="RetrievePostalAttributes">
        <input message="tns:RetrievePostalAttributesRequestMessage"/>
        <output message="tns:RetrievePostalAttributesResponseMessage"/>
    </wsdl:operation>

    <wsdl:operation name="SendMessageToDestination">
        <input message="tns: SendMessageToDestinationRequestMessage"/>
        <output message="tns: SendMessageToDestinationResponseMessage"/>
    </wsdl:operation>

    <wsdl:operation name="SubscribeNotification">
        <input message="tns: SubscribeNotificationRequestMessage"/>
        <output message="tns: SubscribeNotificationResponseMessage"/>
    </wsdl:operation>

    <wsdl:operation name="UnSubscribeNotification ">
```

```
        <input message="tns: UnSubscribeNotificationRequestMessage"/>
        <output message="tns: UnSubscribeNotificationResponseMessage"/>
    </wsdl:operation>

    <wsdl:operation name="ReceiveNotification">
        <input message="tns: ReceiveNotificationRequestMessage"/>
        <output message="tns: ReceiveNotificationResponseMessage"/>
    </wsdl:operation>

    <wsdl:operation name="RejectMessage">
        <input message="tns: RejectMessageRequestMessage"/>
        <output message="tns: RejectMessageResponseMessage"/>
    </wsdl:operation>

</wsdl:portType>

<!-- WSDL schema Binding and Operations elements -->

<wsdl:binding name="PReMServiceBinding" type="tns:PReMServicePortType">
    <soap:binding style="document"
        transport="http://schemas.xmlsoap.org/soap/http"/>

    <wsdl:operation name="Verify">
        <soap:operation soapAction="Verify"/>
            <wsdl:input>
                <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal"/>
            </wsdl:output>
    </wsdl:operation>

    <wsdl:operation name="Encrypt">
        <soap:operation soapAction="Encrypt"/>
            <wsdl:input>
                <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal"/>
            </wsdl:output>
    </wsdl:operation>

    <wsdl:operation name="Decrypt">
        <soap:operation soapAction="Decrypt"/>
            <wsdl:input>
                <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal"/>
            </wsdl:output>
    </wsdl:operation>

    <wsdl:operation name="CheckIntegrity">
        <soap:operation soapAction="CheckIntegrity"/>
            <wsdl:input>
                <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal"/>
            </wsdl:output>
    </wsdl:operation>

    <wsdl:operation name="RetrieveResults">
        <soap:operation soapAction="RetrieveResults"/>
            <wsdl:input>
                <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal"/>
            </wsdl:output>
    </wsdl:operation>

    <wsdl:operation name="Locate">
        <soap:operation soapAction="Locate"/>
            <wsdl:input>
                <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal"/>
            </wsdl:output>
    </wsdl:operation>

    <wsdl:operation name="PostMark">
        <soap:operation soapAction="PostMark"/>
            <wsdl:input>
                <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal"/>
            </wsdl:output>
    </wsdl:operation>

    <wsdl:operation name="Sign">
        <soap:operation soapAction="Sign"/>
            <wsdl:input>
                <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal"/>
            </wsdl:output>
    </wsdl:operation>

    <wsdl:operation name="LogEvent">
        <soap:operation soapAction="LogEvent"/>
            <wsdl:input>
                <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal"/>
            </wsdl:output>
    </wsdl:operation>

    <wsdl:operation name="RetrievePostalAttributes">
        <soap:operation soapAction="RetrievePostalAttributes"/>
            <wsdl:input>
                <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal"/>
            </wsdl:output>
    </wsdl:operation>

    <wsdl:operation name="SendMessageToDestination">
        <soap:operation soapAction=" SendMessageToDestination "/>
            <wsdl:input>
                <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal"/>
            </wsdl:output>
    </wsdl:operation>

    <wsdl:operation name="SubscribeNotification">
        <soap:operation soapAction="SubscribeNotification"/>
```

```
            <wsdl:input>
                <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal"/>
            </wsdl:output>
    </wsdl:operation>

    <wsdl:operation name="UnSubscribeNotification">
        <soap:operation soapAction=" UnSubscribeNotification"/>
            <wsdl:input>
                <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal"/>
            </wsdl:output>
    </wsdl:operation>

    <wsdl:operation name="ReceiveNotification">
        <soap:operation soapAction=" ReceiveNotification"/>
            <wsdl:input>
                <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal"/>
            </wsdl:output>
    </wsdl:operation>

    <wsdl:operation name="RejectMessage">
        <soap:operation soapAction=" RejectMessage"/>
            <wsdl:input>
                <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal"/>
            </wsdl:output>
    </wsdl:operation>

</wsdl:binding>

<!-- WSDL schema Service and Location elements -->

<wsdl:service name="PReMService">
    <wsdl:port name="PReMServicePort" binding="tns:PReMServiceBinding">
        <soap:address location="http://www.myPostalAdmin.com:8000/PReMService"/>
    </wsdl:port>
</wsdl:service>

</wsdl:definitions>
```

Annex C
(informative)

Relevant intellectual property rights (IPR)

C.1 Introduction

This informative annex lists all Intellectual Property Rights (trademarks, patents and patent applications), the use of which has been advised as possibly being implied by the application of this standard. It is stressed that the content of this annex is not exhaustive and is provided on a 'without prejudice' basis. That is:

— mention herein of a particular IPR indicates only that some party has expressed, to the Secretariat of the CEN/TC 331 or the Secretariat of the UPU Standards Board, the view that use of the standard might, in some circumstances, infringe the mentioned right. It should not be taken as in any way confirming the validity of such view and users of this standard should conduct their own searches to determine whether the mentioned IPR is in fact applicable to their specific case;

— the descriptive text associated with mention of a particular IPR is intended to provide only a general indication of the field of application of the right concerned. It should not be taken as implying that the right or its application is limited, in scope, to the description given;

— the absence of reference to any IPR is indicative only of the fact that the Secretariat of the CEN/TC 331 or Secretariat of the UPU Standards Board had, up to the time of publication of this standard, received no suggestion that the said IPR could potentially be infringed by the use of this standard. The UPU or CEN has NOT conducted any patent, trademark or other searches relevant to the subject matter of this standard and cannot accept any responsibility in case of infringement, on the part of users of this document, of any third party intellectual property rights.

Users of the standard are encouraged to conduct any necessary searches and to ensure that any pertinent IPR is either in the public domain; is licensed from the holder(s) or is avoided.

C.2 IPR advised

IPR in respect of which the UPU has been advised that usage might be implied by the use of this Technical Specification:

Bibliography

This bibliography provides full reference and sourcing information for all standards and other reference sources which are quoted in the above text. For references which mention specific version numbers or dates, subsequent amendments to, or revisions of, any of these publications might not be relevant. However, users of this document are encouraged to investigate the existence and applicability of more recent editions. For references without date or version number, the latest edition of the document referred to applies. It is stressed that only referenced documents are listed here.

Internet Engineering Task Force Public Key Infrastructure X.509 working group (IETF PKIX)

NOTE Available at www.ietf.org.

[1] RFC 2822 – *Internet Message Format* (April 2001) P. Resnick, available from: http://www.rfc-editor.org/rfc/pdfrfc/rfc5280.txt.pdf

[2] RFC 3739 – *Internet X.509 Public Key Infrastructure: Qualified Certificates Profile* (March 2004) S. Santesson, M. Nystrom, T. Polk, available from: http://www.rfc-editor.org/rfc/pdfrfc/rfc3739.txt.pdf

European Telecommunications Standards Institute (ETSI) Standards

NOTE ETSI standards are available at www.etsi.org.

[3] ETSI TS 101 733 V1.7.4, July 2008 – *Electronic Signatures and Infrastructures (ESI); CMS Advanced Electronic Signatures (CAdES)*

[4] ETSI TS 101 903 V.1.3.2, March 2006 – *Electronic Signatures and Infrastructures (ESI); XML Advanced Electronic Signatures (XAdES)*

European Union (EU)

[5] *Directive 1999/93/EC of the European Parliament and of the Council of 13 December 1999 on a Community framework for electronic signatures*, OJ L 13, 19.1.2000, p. 12–20, available from: http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2000:013:0012:0020:EN:PDF

This page deliberately left blank