Network Working Group
Request for Comments: 2524
Category: Informational

M. Banan
Neda Communications, Inc.

February 1999

Neda's
Efficient Mail Submission and Delivery (EMSD)
Protocol Specification Version 1.3

This Is A Copy

The Postscript, PDF and HTML formats of RFC-2524 are not the authoritative RFC. The formal, authoritative publication of RFC-2524 is the text format published at the online RFC library.

The author has made every effort to ensure that the Postscript, PDF and HTML formats contain the exact same text as the original. If there are any discrepancies between the Postscript, PDF, HTML formats and the original text version, the original text version is always authoritative.

STATUS OF THIS MEMO

This memo provides information for the Internet community. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Copyright Notice

IESG Note

The protocol specified in this document may be satisfactory for limited use in private wireless IP networks. However, it is unsuitable for general-purpose message transfer or for transfer of messages over the public Internet, because of limitations that include the following:

- Lack of congestion control

  EMSD is layered on ESRO [RFC 2188], which does not provide congestion control. This makes EMSD completely unsuitable for end-to-end use across the public Internet. EMSD should be considered for use in a wireless network only if all EMSD email exchanged between the wireless network and the public Internet will transit an EMSD<->SMTP gateway between the two regions.

- Inadequate security

  The document specifies only clear-text passwords for authentication. EMSD should be used across a wireless network only if sufficiently strong encryption is in use to protect the clear-text password.

- Lack of character set internationalization

  EMSD has no provision for representation of characters outside of the ASCII repertoire or for language tags.

- Poorly defined gatewaying to and from Internet Mail

  Because Internet Mail and EMSD have somewhat different and conflicting service models and different data models, mapping between them may provide good service only in limited cases, and this may cause operational problems.

The IESG therefore recommends that EMSD deployment be limited to narrow circumstances, i.e., only to communicate with devices that have inherent limitations on the length and format of a message (no more than a few hundred bytes of ASCII text), using either:

a. wireless links with adequate link-layer encryption and gatewayed to the public Internet, or

b. a private IP network that is either very over-provisioned or has some means of congestion control.

In the near future, the IESG may charter a working group to define an Internet standards-track protocol for efficient transmission of electronic mail messages, which will be highly compatible with existing Internet mail protocols, and which wil be suitable for operation over the global Internet, including both wireless and wired links.

ABSTRACT

This document specifies the protocol and format encodings for Efficient Mail Submission and Delivery (EMSD). EMSD is a messaging protocol that is highly optimized for submission and delivery of short Internet mail messages. EMSD is designed to be a companion to existing Internet mail protocols.

This specification narrowly focuses on submission and delivery of short mail messages with a clear emphasis on efficiency. EMSD is designed specifically with wireless network (e.g., CDPD, Wireless-IP, Mobile-IP) usage in mind. EMSD is designed to be a natural enhancement to the mainstream of Internet mail protocols when efficiency in mail submission and mail delivery are important. As such, EMSD is anticipated to become an initial basis for convergence of Internet Mail and IP-based Two-Way Paging.

The reliability requirement for message submission and message delivery in EMSD are the same as existing email protocols. EMSD protocol accomplishes reliable connectionless mail submission and delivery services on top of Efficient Short Remote Operations (ESRO) protocols as specified in RFC-2188 [**?**].

Most existing Internet mail protocols are not efficient. Most existing Internet mail protocols are designed with simplicity and continuity with SMTP traditions as two primary requirements. EMSD is designed with efficiency as a primary requirement.

The early use of EMSD in the wireless environment is manifested as IP-based Two-Way Paging services. The efficiency of this protocol also presents significant benefits for large centrally operated Internet mail service providers.

Contents

1 PRELIMINARIES

Mail in the Internet was not a well-planned enterprise, but instead arose in more of an ”organic” way.

This introductory section is not intended to be a reference model and concept vocabulary for mail in the Internet. Instead, it only provides the necessary preliminaries for the concepts and terms that are essential to this specification.

1.1 Internet Mail Submission and Delivery

For the purposes of this specification, mail submission is the process of putting mail into the mail transfer system (MTS).

For the purposes of this specification, mail delivery is the process of the MTS putting mail into a user’s final mail-box.

Throughout the Internet, presently most of mail submission and delivery is done through SMTP.

SMTP was defined as a message *transfer* protocol, that is, a means to route (if needed) and deliver mail by putting finished (complete) messages in a mail-box. Originally, users connected to servers from terminals, and all processing occurred on the server. Now, a split-MUA (Mail User Agent) model is common, with MUA functionality occurring on both the user’s own system and the server.

In the split-MUA model, getting the messages to the user is accomplished through access to a mail-box on the server through such protocols as POP and IMAP. In the split-MUA model, user’s access to its message is a ”Message Pull” paradigm where the user is required to poll his mailbox. Proper message delivery based on a ”Message Push” paradigm is presently not supported. The EMSD protocol addresses this shortcoming with an emphasis on efficiency.

In the split-MUA model, message submission is often accomplished through SMTP. SMTP is widely used as a message *submission* protocol. Widespread use of SMTP for submission is a reality, regardless of whether this is good or bad. EMSD protocol provides an alternative mechanism for message submission which emphasizes efficiency.

1.2 Relationship Of EMSD To Other Mail Protocols

Various Internet mail protocols facilitate accomplishment of various functions in mail processing.

Figure 1, categorizes the capabilities of SMTP, IMAP, POP and EMSD based on the following functions:

- Mail Submission
- Mail Delivery
- Mail Routing (Relay)
- Mail Retrieval

Protocols / Functions	SMTP	IMAP	POP	EMSD
Submission	XX			XXX
Delivery	XXX			XXX
Relay (Routing)	XXX			
Retrieval		XXX	XXX	XX
Mailbox Access		XXX	X	
Mailbox Synch.		XXX		

Figure 1: Messaging Protocols vs. Supported Functions

- Mail-box Access
- Mail-box Synchronization

In Figure 1, the number of "X"es in each box denotes the extent to which a particular function is supported by a particular protocol.

Figure 1 clearly shows that combinations of these protocols can be used to complement each other in providing rich functionality to the user. For example, a user interested in highly mobile messaging functionalities can use EMSD for "submission and delivery of time critical and important messages" and use IMAP for comprehensive access to his/her mail-box.

For mail submission and delivery of short messages EMSD is up to 5 times more efficient than SMTP both in terms of the number of packets transmitted and in terms of number of bytes transmitted. Even with PIPELINING and other possible optimizations of SMTP, EMSD is up to 3 times more efficient than SMTP both in terms of the number of packets transmitted and in terms of number of bytes transmitted. Various efficiency studies comparing EMSD with SMTP, POP and IMAP are available. See Section C.1.1 for more information about comparison of SMTP and EMSD's efficiency.

1.3 EMSD Requirements and Goals

The requirements and goals driving design of EMSD protocol are enumerated below.

1. Provide for submission of short mail messages with the same level of functionality (or higher) that the existing Internet mail protocols provide.
2. Provide for delivery of short mail messages with the same level of functionality (or higher) that the existing Internet mail protocols provide.

3. Function as an extension of the existing mainstream Internet mail.
4. Minimize the number of transmissions.
5. Minimize the number of bytes transmitted.
6. Be quick: minimize latency of message submission and delivery.
7. Provide the same level of reliability (or higher) that the existing email protocols provide.
8. Accommodate varying sizes of messages: the size of a message may determine how the system deals with the message, but the system must accommodate it.
9. Be power efficient and respect mobile platform resources: including memory and CPU levels, as well as battery power longevity (i.e. client-light and server-heavy).
10. Highly extendible: different users will demand different options, so the solution cannot require every feature to be a part of every message. Likewise, usage will emerge that is not currently recognized as a requirement. The solution must be extendible enough to handle new, emerging requirements.
11. Secure: provide the same level of security (or higher) that the existing email protocols provide. Content confidentiality, originator/recipient authentication and message integrity must be available options to users.
12. Easy to implement: Re-use existing technology as much as possible.

1.4 Anticipated Uses Of EMSD

Any network and network operator which has significant bandwidth and capacity limitations can benefit from the use of EMSD. Any network user who must bear high costs for measured network usage can benefit from the use of EMSD.

Initial uses of EMSD is anticipated to be primarily over IP-based wireless networks to provide two-way paging services.

EMSD can also function as an adjunct to Mail Access Protocols for ”Mail Notification Services”.

Considering:

- that most wireless networks shall converge toward being IP-based;
- that two-way paging is the main proven application in most wide-area wireless networks;
- that two-way paging industry and the Internet Email industry can and should converge based on a set of open protocols that address the efficiency requirements adequately;
- that existing Internet email protocols are not bandwidth efficient;
- that existing Internet email protocols do not properly support the ”push” model of delivery of urgent messages,

the EMSD protocol is designed to facilitate the convergence of IP-based two-way paging and Internet email.

Mail submission and delivery take place at the edges of the network. More than one mail submission and delivery protocols which address requirements specific to a particular user's environment are likely to be developed. Such diversity on the edges of the network is desirable and with the right protocols, this diversity does not adversely impact the integrity of the mail transfer system. EMSD is the initial basis for the mail submission and delivery protocol to be used when the user's environment demands efficiency.

1.5 Definitions of Terms Used in this Specification

The following informal definitions and acronyms are intended to help describe EMSD model described in this specification.

Efficient Mail Submission and Delivery Protocol (EMSD-P): The protocol used to transfer messages between the EMSD - Server Agent (e.g., a Message Center) and the EMSD - User Agent (e.g., a Two-Way Pager), see Figure 2.

Message Transfer Agent (MTA)

Message Transfer Service (MTS)

Message Routing Service (MRS): Collection of MTAs responsible for mail routing.

Message User Agent (MUA)

Efficient Mail Submission Server Agent (EMS-SA): An Application Process which conforms to this protocol specification and accepts mail from an EMS-UA and transfers it towards its recipients.

Efficient Mail Delivery Server Agent (EMD-SA): An Application Process which conforms to this protocol specification and delivers mail to an EMD-UA.

Efficient Mail Submission and Delivery Server Agent (EMSD-SA): An Application Process which incorporates both EMS-SA and EMD-SA capabilities.

Efficient Mail Submission User Agent (EMS-UA): An Application Process which conforms to this protocol specification and submits mail to EMS-SA.

Efficient Mail Delivery User Agent (EMD-UA): An Application Process which conforms to this protocol specification and accepts delivery of mail from EMD-SA.

Efficient Mail Submission and Delivery User Agent (EMSD-UA): An Application Process which incorporates both EMS-UA and EMD-UA capabilities.

1.6 Conventions Used In This Specification

The key words "MUST", "MUST NOT", "SHOULD", "SHOULD NOT", and "MAY" in this specification are to be interpreted as defined in [**?**].

This specification uses the ES-OPERATION notation defined in Efficient Short Remote Operations (ESRO) protocols as specified in RFC-2188 [**?**].

Operations and information objects are typically described using the ES-OPERATION and ASN.1 notations in the relevant sections of the specification.

The complete machine verifiable ASN.1 modules are also compiled in one place in Appendix A and Appendix B.

1.7 About This Specification

This protocol specification constitutes a point-of-record. It documents information exchanges and behaviors of existing implementations. It is a basis for implementation of efficient mail submission and delivery user agents and servers.

This specification has been developed entirely outside of IETF. It has had the benefit of review by many outside of IETF. Much has been learned from existing implementations of this protocol. A number of deficiencies and areas of improvement have been identified and are documented in this specification.

This protocol specification is being submitted on October 23, 1998 for timely publication as an Informational RFC.

Future development and enhancements to this protocol may take place inside of IETF.

2 EFFICIENT MAIL SUBMISSION AND DELIVERY OVERVIEW

This section offers a high level view of the Efficient Mail Submission and Delivery Protocol and Format Standards (EMSD-P&FS).

The EMSD-P&FS are used to transfer messages between the EMSD - Server Agent (e.g., a Message Center) and the EMSD - User Agent (e.g., a Two-Way Pager), see Figure 2.

This specification defines the protocols between an EMSD - User Agent (EMSD-UA) and an EMSD - Server Agent (EMSD-SA). The EMSD - P&FS consist of two independent components:

1. EMSD Format Standard (EMSD-FS).

   EMSD-FS is a non-textual form of compact encoding of Internet mail (RFC-822) messages which facilitates efficient transfer of messages. EMSD-FS is used in conjunction with the EMSD-P but is not a general replacement for RFC-822. EMSD-FS defines a method of representation of short interpersonal messages. It defines the "Content" encoding (Header + Body). Although EMSD-FS contains end-to-end information its scope is purely point-to-point. EMSD-FS relies on EMSD-P (see 2 below) for the transfer of the content to its recipients.

   This is described in the section entitled EMSD Format Standards.

2. Efficient Mail Submission and Delivery Protocol (EMSD-P).

   EMSD-P is responsible for wrapping an EMSD-FS message (see 1 above) in a point-to-point envelope and submitting or delivering it. EMSD-P relies on the services of Efficient Short Remote Operation Services (ESROS) as specified in RFC-2188 [**?**] for transporting the point-to-point envelope. Some of

```
+----------------------------------------------+
| MTS                                          |
|                                              |
|  +-------------------------+                 |
|  | MRS                     |                 |
|  |  +---+         +---+    |                 |
|  |  |   |         | M |    |          +---+  |
|  |  |   |<-------->| T |<------------>|   |  |
|  |  |   |         | A |    |          |   |  |               +---+
|  |  |   |         +---+    |          | E |  |               | E |
|  |  |   |                  |          | M |  |               | M |
|  |  | M |                  |          | S |  |   EMSD-P&FS   | S |
|  |  | T |<--------------------------->| D |<---------------->| D |
|  |  | A |                  |          | - |  |               | - |
|  |  |   |         +---+    |          | S |  |               | U |
|  |  |   |         | M |    |          | A |  |               | A |
|  |  |   |<-------->| T |<------------>|   |  |               +---+
|  |  |   |         | A |    |          |   |  |
|  |  +---+         +---+    |          +---+  |
|  |                         |                 |
|  +-------------------------+                 |
|                                              |
|                                              |
+----------------------------------------------+
```

Figure 2: Efficient Mail Submission and Delivery Protocol

the services of EMSD-P include: message originator authentication and optional message segmentation and reassembly. The EMSD-P is expressed in terms of abstract services using the ESROS notation. This is described in the section entitled Efficient Mail Submission and Delivery Protocol.

It is important to recognize that EMSD-P and EMSD-FS are not end-to-end, but focus on the point-to-point transfer of messages. The two points being EMSD-SA and EMSD-UA. EMSD-P function as elements of the Internet mail environment, which provide end-to-end (EMSD-User to any other Messaging Originator or Recipient) services.

Figure 2 illustrates how the EMSD-P&FS defines the communication between a specific EMSD-UA and a specific EMSD-SA. The Message Transfer System may include a number of EMSD-SAs. Each EMSD-SA may have any number of EMSD-UAs with which it communicates.

The Efficient Mail Submission and Delivery Services use the Efficient Short Remote Operation Services (ESROS). They also use the Duplicate Operation Detection Support Functions as described in the section entitled Duplicate Operation Detection Support Functions. These functions guarantee that an operation is performed no more than once.

3 EFFICIENT MAIL SUBMISSION AND DELIVERY PROTOCOL

EM Submission is the process of transferring a message from EMSD-UA to EMSD-SA. EM Delivery is the process of transferring a message from EMSD-SA to EMSD-UA.

The Message-submission service enables an EMSD-UA to submit a message to the EMSD-SA for transfer and delivery to one or more recipients. The Message-submission Service comprises of the submit operation – invoked by the EMSD-UA – and possibly the submitVerify operation – invoked by the EMSD-SA.

The Message-delivery service enables the EMSD-SA to deliver a message to an EMSD-UA. The Message-delivery Service comprises of the deliver operation – invoked by the EMSD-SA – and possibly the deliverVerify operation – invoked by the EMSD-UA.

EMSD-UA uses the following services:

- Message-submission
- Delivery-control (the deliveryControl operation).

EMSD-SA uses the following services:

- Message-delivery
- Submission-control (the submissionControl operation).

This specification expresses information objects using ASN.1 [X.208].

This specification expresses Remote Operations based on the model of ESROS as specified in Efficient Short Remote Operations (RFC-2188) [?]. The ES-OPERATION notation of (RFC-2188) is used throughout this specification to define specific operations.

This specification uses the Duplicate Operation Detection Support functions as specified in Section 4.

3.1 Use Of Lower Layers

3.1.1 Use of ESROS

ESRO protocol, as specified in (RFC-2188 [?]), provides reliable connectionless remote operation services on top of UDP [?] with minimum overhead. ESRO protocol supports segmentation and reassembly, concatenation and separation.

ESRO Services (2-Way and 3-Way handshake) shall be used by the EMSD-P.

ESRO Service Access Point (SAP) selectors used by EMSD-P are enumerated in the protocol.

3.1.2 Use Of UDP

EMSD-P through ESRO MUST use UDP [?] port number 642 (esro-emsdp).

Note that specification of Service Access Points (SAP) for EMSD-P include the UDP Port Number specification in addition to ESRO SAP selector specifications. In other words, EMSD-P's use of ESRO SAPs does not preclude use of the same SAP selectors by other protocols which use a UDP port other than port 642. Such usage of ESRO is a design characteristic of ESRO which results into bandwidth efficiency and is not a scalability limitation.

3.1.3 Encoding Rules

Use of Basic Encoding Rules (BER) [?] is mandatory for both EMSD Format Standards and EMSD Protocol.

In order to minimize data transfer, the following restrictions shall be maintained in the formatting of EMSD PDUs:

- Specifically, when ASN.1 Basic Encoding Rules are being used:
  A. Only the "Definite" form of Length encoding MUST be used,
  B. The "Short" form of Length encoding MUST be used whenever possible (i.e. when the Length is less than 128), and
  C. OCTET STRING and BIT STRING values, and any other native ASN.1 types which may be encoded as either "Primitive" or "Constructed", MUST always be encoded as "Primitive" and MUST never be "Constructed".

3.1.4 Presentation Context

Parameter Encoding Type of "0" MUST be used in ESRO Protocol to identify Basic Encoding Rules for operation arguments.

3.2 EMSD-UA Invoked Operations

The following operations are invoked by EMSD-UA:

a. submit
b. deliveryControl
c. deliveryVerify

The submit operation uses the duplication detection functional unit while deliveryControl and deliveryVerify don't use the duplication detection.

The complete definition of these operations follows.

3.2.1 submit

The submit ES-OPERATION enables an EMSD-UA to submit a message to the EMSD-SA for transfer and delivery to one or more recipients.

```
submit ES-OPERATION

    ARGUMENT SubmitArgument
    RESULT SubmitResult
    ERRORS
    {
        submissionControlViolated,
        securityError,
        resourceError,
        protocolViolation,
        messageError
    } ::= 33;
```

Duplicate operation detection is necessary for this operation.

The successful completion of the ES-OPERATION signifies that the EMSD-SA has accepted responsibility for the message (but not that it has delivered it to its intended recipients).

The disruption of the ES-OPERATION by an error signifies that the EMSD-SA cannot assume responsibility for the message.

Arguments

This operation's arguments are:

```
SubmitArgument ::= SEQUENCE
{
  -- Security features
  security                [0]    IMPLICIT SecurityElement OPTIONAL,

  -- Segmentation features for efficient transport
  segment-info                          SegmentInfo OPTIONAL,

  -- Content type of the message
  content-type                          ContentType,

  --
  -- THE CONTENT --
  --

  -- The submission content
  content                               ANY DEFINED BY content-type
```

```
};
```

The fields are:

Security

See Section 3.4.1, ”SecurityElements”.

Segment-info

See Section 3.4.2, ”Message Segmentation and Reassembly”.

Content-type

This argument identifies the type of the content of the message. It identifies the abstract syntax and the encoding rules used.

Content

This argument contains the information the message is intended to convey to the recipient(s). It shall be generated by the originator of the message.

Results

This operation’s results are:

```
SubmitResult ::= SEQUENCE

    {
        -- Permanent identifier for this message.
        -- Also contains the message submission time.
        -- See comment regarding assignment of message identifiers,
        -- at the definition of EMSDLocalMessageId.

        message-id                              EMSDLocalMessageId
    };
```

The fields are:

Message-id

This result contains an EMSD-SA-identifier that uniquely and unambiguously identifies the message-submission. It shall be generated by the EMSD-SA.

Errors

See Section 3.4.3.

3.2.2 deliveryControl

The deliveryControl ES-OPERATION enables the EMSD-UA to temporarily limit the operations that the EMSD-SA may invoke, and the messages that the EMSD-SA may deliver to the EMSD-UA via the Message delivery ES-OPERATION.

```
deliveryControl ES-OPERATION
    ARGUMENT DeliveryControlArgument
    RESULT DeliveryControlResult
    ERRORS
    {
        securityError,
        resourceError,
        protocolViolation
    } ::= 2;
```

The duplicate operation detection is not required for this operation.

The EMSD-SA shall hold until a later time, rather than abandon, ES-OPERATIONS and messages that are presently suspended.

The successful completion of the ES-OPERATION signifies that the specified controls are now in force.

The ES-OPERATION returns an indication of any ES-OPERATIONS that the EMSD-SA would invoke, or any message types that the EMSD-SA would deliver, were it not for the prevailing controls.

Arguments

This operation's arguments are:

```
DeliveryControlArgument ::= SEQUENCE
{
  -- Request an addition of or removal of a set of restrictions
```

```
  restrict                [0]     IMPLICIT Restrict DEFAULT update,

  -- Which operations are to be placed in the restriction set
  permissible-operations  [1]     IMPLICIT Operations OPTIONAL,

  -- What maximum content length should be allowed
  permissible-max-content-length

                                  [2]     IMPLICIT INTEGER
                                   (0..ub-content-length) OPTIONAL,

  -- What is the lowest priority message which may be delivered
  permissible-lowest-priority

                                  [3]     IMPLICIT ENUMERATED
                                           {
                                             non-urgent     (0),
                                             normal         (1),
                                             urgent         (2)
                                           } OPTIONAL,

  -- Security features
  security                        [4]     IMPLICIT SecurityElement
                                          OPTIONAL,

  -- User Feature selection
  user-features                   [5]     IMPLICIT OCTET STRING
                                          OPTIONAL
};
```

Restrict

This argument indicates whether the controls on ES-OPERATIONS are to be updated or removed. It may be generated by the EMSD-UA.

This argument may have one of the following values:

- update: The other arguments update the prevailing controls;
- remove: All temporary controls are to be removed

In the absence of this argument, the default update shall be assumed.

Permissible-operations

This argument indicates the ES-OPERATIONS that the EMSD-SA may invoke on the EMSD-UA. It may be generated by the EMSD-UA.

This argument may have the value allowed or prohibited for each of the following:

- message-delivery: The EMSD-SA may/may not invoke the deliver ES-OPERATIONS; and
- Other ES-OPERATIONS are not subject to controls, and may be invoked at any time.

In the absence of this argument, the ES-OPERATIONS that the EMSD-SA may invoke on the EMSD-UA are unchanged.

Permissible-max-content-length

This argument contains the content-length, in octets, of the longest-content message that the EMSD-SA shall deliver to the EMSD-UA via the deliver ES-OPERATIONS. It may be generated by the EMSD-UA.

In the absence of this argument, the permissible-maximum-content-length of a message that the EMSD-SA may deliver to the EMSD-UA is unchanged.

Permissible-lowest-priority

This argument contains the priority of the lowest priority message that the EMSD-SA shall deliver to the EMSD-UA via the deliver ES-OPERATIONS. It may be generated by the EMSD-UA.

This argument may have one of the following values of the priority argument of the submit ES-OPERATIONS: normal, non-urgent or urgent.

In the absence of this argument, the priority of the lowest priority message that the EMSD-SA shall deliver to the EMSD-UA is unchanged.

Security

See Section 3.4.1, ”SecurityElements”.

User-features

This argument contains information that allows the EMSD-UA to convey to MTS the feature set that the user is capable of supporting. This argument will be defined when the setConfiguration and getConfiguration operations are defined.

Results

```
DeliveryControlResult ::= SEQUENCE
```

```
{
  -- Operation types queued at the EMSD-SA due to existing
  -- restrictions.
  waiting-operations      [0]     IMPLICIT Operations DEFAULT { },

  -- Types of messages queued at the EMSD-SA due to
  -- existing restrictions
  waiting-messages        [1]     IMPLICIT WaitingMessages
                                  DEFAULT { },

  -- Content Types of messages queued at the EMSD-SA
  waiting-content-types   SEQUENCE SIZE (0..ub-content-types) OF
                                         ContentType DEFAULT { }

};

Restrict ::= ENUMERATED
{
    update                                      (1),
    remove                                      (2)
};

Operations ::= BIT STRING
{
    submission                                  (0),
    delivery                                    (1)
};

WaitingMessages ::= BIT STRING
{
    long-content                                (0),
    low-priority                                (1)
};
```

Waiting-operations

This result indicates the ES-OPERATIONS being held by the EMSD-SA, and that the EMSD-SA would invoke on the EMSD-UA if it were not for the prevailing controls. It may be generated by the EMSD-SA.

This result may have the value holding or not-holding for each of the following:

- message-delivery: The EMSD-SA is/is not holding messages, and would invoke the deliver ES-OPERATIONS on the EMSD-UA if it were not for the prevailing controls.

In the absence of this result, it may be assumed that the EMSD-SA is not holding any messages for delivery due to the prevailing controls.

Waiting-messages

This result indicates the kind of messages the EMSD-SA is holding for delivery to the EMSD-UA, and would deliver via the deliver ES-OPERATIONS, if it were not for the prevailing controls. It may be generated by the EMSD-SA.

This result may have one or more of the following values:

- long-content: The EMSD-SA has messages held for delivery to the EMSD-UA which exceed the permissible maximum-content-length control currently in force;
- low-priority: The EMSD-SA has messages held for delivery to the EMSD-UA of a lower priority than the permissible-lowest-priority control currently in force;

In the absence of this result, it may be assumed that the EMSD-SA is not holding any messages for delivery to the EMSD-UA due to the permissible-maximum-content- length, permissible-lowest-priority or permissible-security context controls currently in force.

Errors

See Section 3.4.3.

3.2.3 deliveryVerify

The deliveryVerify ES-OPERATIONS enables the EMSD-UA to verify delivery of a message when it receives FAILURE.indication for deliver ES-OPERATIONS.

```
deliveryVerify ES-OPERATION

    ARGUMENT DeliveryVerifyArgument
    RESULT DeliveryVerifyResult
    ERRORS
    {
        verifyError,
        resourceError,
        protocolViolation
    } ::= 5;
```

The duplicate operation detection is not required for this operation.

Arguments

This operation's arguments are:

```
DeliveryVerifyArgument ::= SEQUENCE

{
  -- Identifier of this message. This is the same identifier that
  -- was provided to the originator in the Submission Result.
  -- See comment regarding assignment of message identifiers,
  -- at the definition of EMSDMessageId.
  message-id                                      EMSDMessageId
};
```

Message-id

This argument contains an EMSD-SA-identifier that distinguishes the message from all other messages. It shall be generated by the EMSD-SA, and shall have the same value as the message-submission-identifier supplied to the originator of the message when the message was submitted.

Results

```
DeliveryVerifyResult ::= SEQUENCE
{
         status  DeliveryStatus
};

 DeliveryStatus  ::= ENUMERATED
{
        no-report-is-sent-out                     (1),
        delivery-report-is-sent-out               (2),
        non-delivery-report-is-sent-out           (3)
 };
```

No-report-is-sent-out

This result indicates that EMSD-SA has received the delivery verify and no report is sent out (either because it has not been requested or EMSD-SA has problems and can not send it out).

Delivery-report-is-sent-out

This result indicates that EMSD-SA has received the delivery verify and has sent the delivery report out.

Non-Delivery-report-is-sent-out

This result indicates that EMSD-SA has received the delivery verify but it has already sent out a non-Delivery report. This should not happen in normal cases but a wrong user profile on EMSD-SA side can result in this outcome.

Errors

See Section 3.4.3.

3.3 EMSD-SA Invoked Operations

This section defines the operations invoked by the EMSD-SA:

a. deliver;

b. submissionControl;

c. submissionVerify.

The deliver operation uses 3-Way handshake service of ESROS. This operation always uses the duplication detection functional unit.

The submissionControl and submissionVerify operations use 2-Way handshake service of ESROS without duplication detection.

3.3.1 deliver

The deliver ES-OPERATIONS enables the EMSD-SA to deliver a message to an EMSD-UA.

```
deliver ES-OPERATION

    ARGUMENT DeliverArgument
    RESULT NULL
    ERRORS
    {
        deliveryControlViolated,
        securityError,
        resourceError,
        protocolViolation,
        messageError
    } ::= 35;
```

The EMSD-UA MUST not refuse performing the deliver ES-OPERATION unless the delivery would violate the deliveryControl restrictions then in force.

Arguments

This operation's arguments are:

```
DeliverArgument ::= SEQUENCE
{
  -- Identifier of this message. This is the same identifier that
  -- was provided to the originator in the Submission Result.
  -- See comment regarding assignment of message identifiers,
  -- at the definition of EMSDMessageId.
  message-id                                      EMSDMessageId,

  -- Time the message was delivered to the recipient by EMSD-SA
  message-delivery-time                           DateTime,

  -- Time EMSD-SA originally took responsibility for processing
  -- of this message. This field shall be omitted if the message-id
  -- contains an EMSDLocalMessageId, because that field contains
  -- the submission time within it.
  message-submission-time [0]  IMPLICIT DateTime OPTIONAL,

  -- Security features
  security                [1]  IMPLICIT SecurityElement OPTIONAL,

  -- SegContentTypementation features for efficient transport
  segment-info                            SegmentInfo OPTIONAL,

  -- The type of the content
  content-type                              ContentType,

  --
  -- THE CONTENT --
  --

  -- The submitted (and now being delivered) content
  content                          ANY DEFINED BY content-type
};
```

message-id

This argument contains an EMSD-SA-identifier that distinguishes the message from all other messages. When within the EMSD, it MUST be generated by the EMSD-SA, and MUST have the same value as the message-submission-identifier supplied to the originator of the message when the message was submitted.

Message-delivery-time

This argument contains the Time at which delivery occurs and at which the EMSD-SA is relinquishing responsibility for the message. It shall be generated by the EMSD-SA.

Results

This operation returns an empty result as indication of success.

Errors

See Section 3.4.3.

3.3.2 submissionControl

```
submissionControl ES-OPERATION
    ARGUMENT SubmissionControlArgument
    RESULT SubmissionControlResult
    ERRORS
    {
        securityError,
        resourceError,
        protocolViolation
    } ::= 4;
```

The submissionControl ES-OPERATIONS enables the EMSD-SA to temporarily limit the operations that the EMSD-UA may invoke, and the messages that the EMSD-UA may submit to the EMSD-SA via the submit ES-OPERATIONS.

The duplicate operation detection is not required for this operation.

The EMSD-UA should hold until a later time, rather than abandon, ES-OPERATIONS and messages that are presently suspended.

The successful completion of the ES-OPERATIONS signifies that the specified controls are now in force. These controls supersede any previously in force, and remain in effect until the association is released or the EMSD-SA re-invokes the submissionControl ES-OPERATIONS.

The ES-OPERATIONS returns an indication of any ES-OPERATIONS that the EMSD-UA would invoke were it not for the prevailing controls.

Arguments

This operation's arguments are:

```
SubmissionControlArgument ::= SEQUENCE
{
  -- Request an addition of or removal of a set of restrictions
  restrict               [0]     IMPLICIT Restrict DEFAULT update,

  -- Which operations are to be placed in the restriction set
  permissible-operations  [1]     IMPLICIT Operations OPTIONAL,

  -- What maximum content length should be allowed
  permissible-max-content-length
                          [2]     IMPLICIT INTEGER
                                  (0..ub-content-length) OPTIONAL,

  -- Security features
  security                [3]     IMPLICIT SecurityElement
                                                OPTIONAL
};
```

Restrict

This argument indicates whether the controls on ES-OPERATIONS are to be updated or removed. It may be generated by the EMSD-SA.

This argument may have one of the following values:

- update: The other arguments update the prevailing controls;
- remove: All temporary controls are to be removed

In the absence of this argument, the default update shall be assumed.

Permissible-operations

This argument indicates the ES-OPERATIONS that the EMSD-UA may invoke on the EMSD-SA. It may be generated by the EMSD-SA.

This argument may have the value allowed or prohibited for each of the following:

- submit: The EMSD-UA may/may not invoke the submit ES-OPERATIONS; and

- Other ES-OPERATIONS are not subject to controls, and may be invoked at any time.

In the absence of this argument, the ES-OPERATIONS that the EMSD-UA may invoke on the EMSD-SA are unchanged.

Permissible-max-content-length

This argument contains the content-length, in octets, of the longest-content message that the EMSD-UA shall submit to the EMSD-SA via the submit ES-OPERATIONS. It may be generated by the EMSD-SA.

In the absence of this argument, the permissible-maximum-content-length of a message that the EMSD-UA may submit to the EMSD-SA is unchanged.

Security

See Section 3.4.1, ”SecurityElements”.

Results

```
SubmissionControlResult ::= SEQUENCE
{
  -- Operation types queued at the EMSD-SA due to existing
  -- restrictions.
  waiting-operations    [0]   IMPLICIT Operations DEFAULT { }

};
```

Waiting-operations

This result indicates the ES-OPERATIONS being held by the EMSD-UA, and that the EMSD-UA would invoke if it were not for the prevailing controls. It may be generated by the EMSD-UA.

This result may have the value holding or not-holding for each of the following:

- submit: The EMSD-UA is/is not holding messages, and would invoke the submit ES-OPERATIONS if it were not for the prevailing controls.

In the absence of this result, it may be assumed that the EMSD-UA is not holding any messages for submission due to the prevailing controls.

Errors

See Section 3.4.3.

3.3.3 submissionVerify

The submissionVerify ES-OPERATIONS enables the EMSD-SA to verify if the EMSD-UA has received the result of its submission.

```
submissionVerify  ES-OPERATION

    ARGUMENT SubmissionVerifyArgument
    RESULT SubmissionVerifyResult
    ERRORS
    {
        submissionVerifyError,
        resourceError,
        protocolViolation
    } ::= 6;
```

The duplicate operation detection is not required for this operation.

Arguments

This operation's arguments are:

```
SubmissionVerifyArgument ::= SEQUENCE

  -- Identifier of this message. This is the same identifier that
  -- was provided to the originator in the Submission Result.
  -- See comment regarding assignment of message identifiers,
  -- at the definition of EMSDMessageId.
  {
     message-id                                  EMSDMessageId
  };
```

Message-id

This argument contains an EMSD-SA-identifier that distinguishes the message from all other messages. It shall be generated by the EMSD-SA, and shall have the same value as the message-submission-identifier supplied to the originator of the message when the message was submitted.

Results

```
SubmissionVerifyResult ::= SEQUENCE
{
        status  SubmissionStatus
};

SubmissionStatus::= ENUMERATED
{
        send-message            (1),
        drop-message            (2)
};
```

Send-message

This result indicates that EMSD-SA is supposed to send the message out.

Drop-message

This result indicates that EMSD-SA is supposed to drop the message.

Errors

See Section 3.4.3.

3.4 EMSD Common Information Objects

3.4.1 SecurityElements

```
SecurityElement ::= SEQUENCE

{
  credentials                          Credentials,
  contentIntegrityCheck                ContentIntegrityCheck OPTIONAL
};

Credentials ::= CHOICE
{
  simple                          [0]     IMPLICIT SimpleCredentials

  -- Strong Credentials are for future study
  -- strong                       [1]     IMPLICIT StrongCredentials
```

```
  -- externalProcedure            [2]     EXTERNAL
};

SimpleCredentials ::= SEQUENCE
{
  eMSDAddress                     EMSDAddress OPTIONAL,
  password                [0]     IMPLICIT OCTET STRING
                          SIZE (0..ub-password-length)) OPTIONAL
};

-- StrongCredentials ::= NULL
-- for now.
-- ContentIntegrityCheck is a 16-bit checksum of content
ContentIntegrityCheck ::= INTEGER (0..65535);
```

3.4.2 Message Segmentation and Reassembly

Small messages can benefit from the efficiencies of connectionless feature of ESROS (See Efficient Short Remote Operations, RFC-2188 [?]).

Very large messages are transferred using protocols (e.g., SMTP) that rely on Connection Oriented Transport Service (e.g., TCP).

When a message is too large to fit in a single connectionless PDU but is not large enough to justify the overhead of connection establishment, it may be more efficient for the message to be segmented and reassembled while the connectionless service of ESROS is used. If the underlying Remote Operation Service is capable of efficient segmentation/reassembly over connectionless (CL) services, then use of the segmenting/reassembly mechanism introduced in this section is not necessary. This feature is accommodated in this layer by:

```
SegmentInfo ::= CHOICE

{
  first           [APPLICATION 2]         IMPLICIT FirstSegment,
  other           [APPLICATION 3]         IMPLICIT OtherSegment
};

FirstSegment ::= SEQUENCE
{
  sequence-id                             INTEGER,
  number-of-segments                      INTEGER
  -- number-of-segments must not exceed ub-total-number-of-segments
};

OtherSegment ::= SEQUENCE
{
  sequence-id                             INTEGER,
  segment-number                          INTEGER
};
```

Segmentation and reassembly only applies to Message-submission and Message-delivery.

The sender of the message is responsible for segmenting the message content into segments that fit in CL PDUs. The segmented content is sent in a sequence of message- segments each carrying a segment of the content. sequence-Id is a unique identifier that is present in all message-segments. In addition to sequence identifier, the first message- segment specifies the total number of segments (number-of-segments). Other message- segments have a segment sequence number (segment-number). The receiver is responsible for sequencing (based on segment-number) and reassembling the entire message.

Segmenting over the Connectionless ESRO Service

The sender of the message maps the original message into an ordered sequence of message-segments. This sequence shall not be interrupted by other messages over the same ESROS association.

All message-segments in the sequence shall be assigned a sequence identifier by sender. The sequence identifier shall be incremented by one by the sender after transmission of a complete message sequence.

The first message-segment specifies the total number of segments. All message- segments in the sequence except the first one shall be sequentially numbered, starting at 1 (first message-segment has implicit segment number of 0).

Each message-segment is transmitted by issuing a Message-submission or Message-delivery ES-OPERATIONS. All segments of a segmented message are identified by the same sequence-id. For a given message, the receiver should not impose any restriction on the order of arrival of message-segments.

There is no requirement that any message-segment content be of maximum length allowed by ESROS for connectionless transmission; however, no more than ub-total-number-of-segments segments can be derived from a single message.

The receiver reassembles a sequence of message-segments into a single message. A message shall not be further processed unless all segments of the message are received. Failure to receive the message shall be determined by the following events:

- Expiration of Reassembly Timer (see Section 3.4.3).
- Receipt of a message-segment with different sequence identifier.

In the event of the above mentioned failures, the receiver shall discard a partially assembled sequence.

In Reassembly process, all arguments other than content are ignored in all segments except the first one. The content parts of all segments are concatenated to compose the original message content.

When sender receives FAILURE.indication (as opposed to a resourceError) for a message-segment, the whole message shall be retransmitted.

In the case of submission and delivery operations, the verify function is used as described below:

Receiver ignores FAILURE.indications received for message-segments, and just collects the message-segments to complete the message. However, it keeps a failure status for a segmented message which says if any segment of the message has received FAILURE.indication. When receiver succeeds to assemble the whole segmented message, then if the status of the message shows there has been a FAILURE.indication for any of the message-segments, it verifies the message through verify operation. It's not enough to invoke verify operation just based on the last message-segment because the sender might send a segment without waiting for the result of the previous segment. In such cases, there might be any combination of success and failure for message- segments on the sender side.

Receiver uses the error code ResourceError (see Section 3.4.3) to ask for retransmission of a single segment and uses the error code MessageError (see Section 3.4.3) to ask for retransmission of all segments (the whole message).

Reassembly Timer

The Reassembly Timer is a local timer maintained by the receiver of message-segments that assists in performing the reassembly function. This timer determines how long a receiver waits for all segments of a message-segment sequence to be received. The timer protects the receiver from the loss of a series of segments and possible sequence identifier wrap-around.

The Reassembly Timer shall be started on receipt of a message-segment with different sequence identifier than that previously received. The timer shall be stopped on receipt of all segments composing the sequence.

The value of Reassembly Timer is defined based on the network characteristics and the number of segments. This requires that the transmission of all segments of a single message must be completed within this time limit.

3.4.3 Common Errors

```
protocolVersionNotRecognized  ERROR PARAMETER NULL ::= 1;

submissionControlViolated  ERROR PARAMETER NULL ::= 2;

messageIdentifierInvalid  ERROR PARAMETER NULL ::= 3;

securityError ERROR PARAMETER security-problem SecurityProblem ::= 4;

deliveryControlViolated   ERROR PARAMETER NULL ::= 5;

resourceError  ERROR PARAMETER NULL ::= 6;

protocolViolation  ERROR PARAMETER NULL ::= 7;

messageError  ERROR PARAMETER NULL ::= 8;

SecurityProblem ::= INTEGER (0..127);
```

protocolVersionNotRecognized

The major and minor protocol versions presented do not match those recognized as being valid.

submissionControlViolated

The Submission control violated error reports the violation by the MTS-user of a control on submission services imposed by the MTS via the Submission control service. The Submission control violated abstract-error has no parameters.

messageIdentifierInvalid

The Message Identifier Invalid error reports that the Message Identifier presented to the MTS is not considered valid.

securityError

The Security error reports that the requested operation could not be provided by the MTS or MTS-user because it would violate the security policy in force.

deliveryControlViolated

The Delivery control violated error reports the violation by the MTS of a control on delivery operations imposed by the MTS-user via the Delivery-control operation.

resourceError

The messaging agent cannot currently support this operation. In the case of segmentation and reassembly, resourceError is by the receiver used to request that the sender retransmit of a single segment.

protocolViolation

Indicates that one or more mandatory argument(s) were missing.

messageError

For a multi-segment message, this error indicates that the messaging agent has not received the message completely and that the message must be retransmitted.

SecurityProblem

To ensure the security-policy is not violated during delivery, the message-security-label is checked against the security-context. If delivery is barred by the security -policy then, subject to the security policy, a report instruction for this is generated.

3.4.4 ContentType

```
ContentType ::=  INTEGER
{
  -- Content type 0 is reserved and shall never be transmitted.
  reserved                                (0),
  -- Content types between 1 and 31 (inclusive) are for
  -- internal-use only
  probe                                   (1), -- reserved
  delivery-report                         (2), -- reserved

  -- Content types between 32 and 63 (inclusive) are for
  -- message types  defined within this specifications.
  emsd-interpersonal-messaging-1995       (32),
  voice-messaging                         (33) -- reserved

  -- Content types beyond and including 64 are for
  -- bilaterally-agreed use between peers.
} (0..127);
```

3.4.5 EMSDMessageId

If this message was originated as an RFC-822 message, then this EMSDMessageId shall be the “Message-Id:” field from that message. If this message was originated within the EMSD domain, then this identifier shall be unique for the EMSD-SA generating this id.

```
EMSDMessageId ::= CHOICE
{
  EMSDLocalMessageId  [APPLICATION 4]
                      IMPLICIT EMSDLocalMessageId,

  rfc822MessageId     [APPLICATION 5]
                      IMPLICIT AsciiPrintableString
                      (SIZE (0..ub-message-id-length))
};

EMSDLocalMessageId ::= SEQUENCE
{
  submissionTime             DateTime,
  messageNumber              INTEGER (0..ub-local-message-nu)
```

```
};
```

3.4.6 EMSDORAddress

```
EMSDORAddress ::= CHOICE
{
  -- This is the local-format address
  emsd-local-address-format           EMSDAddress,

  -- This is a globally-unique RFC-822 Address
  rfc822DomainAddress                 AsciiPrintableString
};
```

In the global sense Originators and Recipients are represented by EMSDORAddress. The rfc822Domain may be used to address any recipient.

3.4.7 EMSDAddress

```
EMSDAddress ::= SEQUENCE
{
  emsd-address        OCTET STRING (SIZE
                      (1..ub-emsd-address-length)),

  -- emsd-address is a decimal integer in BCD
     (Binary Encoded Decimal) format.
  -- If it had an odd number of digits, it is
  -- padded with 0 on the left.

  emsd-name          [0]  IMPLICIT OCTET STRING
                          (SIZE (0..ub-emsd-name-length))
                          OPTIONAL
};
```

Originator and Recipients in the scope of EMSD network are identified by a digit based addressing scheme. EMSDAddress can only be used where the scope of addressing has clearly been limited to the EMSD network.

3.4.8 DateTime

```
DateTime ::= INTEGER;
```

DateTime is a Julian date, expressed as the number of seconds since 00:00:00 UTC, January 1, 1970.

3.4.9 AsciiPrintableString

```
Iso8859String ::=  GeneralString;

AsciiPrintableString ::= [APPLICATION 0]
                         IMPLICIT Iso8859String (FROM

    (" "|"!"|"#"|"$"|"%"|"&"|"'"|"("|")"|"*"|"+"|","|"-"|"."|"/"|
     "0"|"1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9"|":"|";"|"<"|"="|">"|
     "?"|"@"|"A"|"B"|"C"|"D"|"E"|"F"|"G"|"H"|"I"|"J"|"K"|"L"|"M"|
     "N"|"O"|"P"|"Q"|"R"|"S"|"T"|"U"|"V"|"W"|"X"|"Y"|"Z"|"["|"]"|
     "^"|"_"|"`"|"a"|"b"|"c"|"d"|"e"|"f"|"g"|"h"|"i"|"j"|"k"|"l"|
     "m"|"n"|"o"|"p"|"q"|"r"|"s"|"t"|"u"|"v"|"w"|"x"|"y"|"z"|"{"|
     "|"|"}"|"~"|"\"|""""));
```

3.4.10 ProtocolVersionNumber

```
ProtocolVersionNumber ::= [APPLICATION 1]    SEQUENCE
{
  version-major                   INTEGER,
  version-minor           [0]     IMPLICIT INTEGER DEFAULT 0
}
```

3.5 Submission and Delivery Procedures

Table 1 provides a comprehensive summary of EMSD-P operations, the SAP selectors used and the operation IDs used.

Submission

The semantics of a submission operation is Exactly Once. Exactly Once means that every operation is carried out exactly one time, no more and no less. This semantic can not be fully implemented because, if after invoking the operation, an invoker has a Success (e.g. result) indication and the performer has a FAILURE.indication, and the network goes down, the result of the operation will be Zero (and not Exactly Once).

No more than one is controlled and guaranteed by the performer by using the Duplicate Operation Detection Support Functions (see the chapter entitled *Duplicate Operation Detection Support*).

Not zero but one is realized by performer by using the SubmissionVerify operation. When the performer receives FAILURE.indication, it's responsibility is to resolve the case by using SubmissionVerify resulting in Not zero but one.

Submission procedure is as follows:

Operation	Invoker	Sap Sel	Performer	Sap Sel	Duplicate Detect	OpId	ESROS Use
submit	UA	4	MTS	5	Yes	33	3-Way
deliver	MTS	2	UA	3	Yes	35	3-Way
deliveryControl	UA	8	MTS	9	No	2	2-Way
submissionControl	MTS	6	UA	7	No	4	2-Way
submissionVerify	MTS	6	UA	7	No	6	2-Way
deliveryVerify	UA	8	MTS	9	No	5	2-Way
getConfiguration	UA	8	MTS	9	No	7	2-Way
setConfiguration	MTS	6	UA	7	No	8	2-Way

Table 1: EMSD-P Operations Summary

- Submit operation with 3-Way handshake and Duplicate Operation Detection Support Function is invoked.
- If performer at EMSD-SA receives FAILURE.indication, it invokes SubmissionVerify.
- Message is sent out by EMSD-SA only if result operation is confirmed or the operation is verified (in the case of FAILURE.indication).

The semantic of SubmissionVerify operation is At Least Once. This type of semantics corresponds to the case that invoker keeps trying over and over until it gets a proper reply. This operation can be performed more than once without any harm.

Implications:

- MTS sends out the message if and only if it's sure that UA knows about it.

Delivery

The semantics of Deliver operation is Exactly Once. Exactly Once means that every operation is carried out exactly one time, no more and no less. This semantic can not be fully implemented and if after invoking the operation, invoker has Success indication and performer has FAILURE.indication, and the network goes down, the result of the operation will be Zero (and not Exactly Once).

No more than one is controlled and guaranteed by performer and by using the *Duplicate Operation Detection Support Functions*.

Not zero but one is realized by performer by using the DeliveryVerify operation. When performer receives FAILURE.indication, it's responsible to resolve the case by using DeliveryVerify resulting in Not zero but one.

Delivery procedure is as follows:

- Deliver operation with 3-Way handshake is invoked.
- If performer at User Agent (device) receives FAILURE.indication, it invokes DeliveryVerify.

The semantic of DeliveryVerify operation is At Least Once. This type of semantics corresponds to the case that invoker keeps trying over and over until it gets a proper reply. This operation can be performed more than once without any harm.

Implications:

- A non-delivery report is sent by MTS only if the message is not delivered.
- The UA is responsible for notifying the MTS (through an explicit deliveryVerify) to make sure that a delivery report is sent out.

4 DUPLICATE OPERATION DETECTION SUPPORT

4.1 Duplicate Operation Detection Support Overview

Some operations are idempotent in nature, i.e. they can be performed more than once without any harm. However, some other operations are non-idempotent in nature, i.e. they should be performed only once. In the case of non-idempotent operations, performer should be able to detect duplicate operations and perform each non- idempotent operation only once.

Examples of non-idempotent operations are Submission and Delivery of messages which shouldn't be performed more than once. Examples of idempotent operations are Submission-control and Delivery-control which can be performed more than once with no harm.

ESRO Services don't detect duplicate invocation of operations. As a result, the Duplicate Operation Detection Support Functional Unit is used to detect duplication when the same operation instance is invoked more than once. Invoker assigns an Operation Instance Identifier to an operation and this Operation Instance Identifier is used at the peer performer entity to detect the duplicate invocation of the same operation.

Using this support, non-idempotent operations can be repeated over and over with no harm because the duplicate invocations are detected by this functional unit. This support helps the performer not to perform an operation more than once.

Support for duplication detection is realized through allocating Operation Instance Id (see Section 4.1.2, "Operation Instance Identifier") to an operation by invoker. When an operation is invoked using duplication detection support, performer logs the Operation Instance Identifier and checks the next operations against duplication.

Operation value identifies whether performer should detect duplicate operations (see Section 4.1.1, “Operation Value”) and Operation Instance Id is assigned by invoker and sent as the first byte of operation’s parameter.

4.1.1 Operation Value

Operation Values are divided into two groups. Operation values from 0 to 31 do not have Duplicate Operation Detection Support (0 to 31) and operation values from 32 to 63 have Duplicate Operation Detection Support.

Duplicate Operation Detection Functional Unit checks for duplication only if Operation Value is in the range of 32 to 63.

When invoker user uses an Operation Value in the range of 32 to 63 which means operation with support for duplication detection, the user should specify an Operation Instance ID for the operation (see next section).

4.1.2 Operation Instance Identifier

To support duplication detection, an Operation Instance Identifier is assigned by invoker user and sent as the first byte of the operation’s parameter. This identifier is used on performer side to detect duplicate invocation of the same operation. Characteristics of Operation Instance Identifier is as follows:

- Operation Instance Identifier is one byte and can have values from 0 to 255.
- Operation Instance Identifier is sent as the first byte of the operations parameter (without encoding).
- The length of Operation Instance Identifier is 8-bit, but depending on the performer capabilities, it might keep 0 to 127 Operation Instance Identifiers for duplication detection. The performer profile defines the number of outstanding Operation Instance Identifiers that are checked against duplication. When a performer profile indicates support for 0 outstanding Operation Instance Identifier, it means it does not have support for Duplicate Operation Detection. In this case, there should be only one outstanding operation at any point of time.
- Instance ID check is not part of ESROS, per se. Use of Duplicate Detection is determined by EMSD-P. Operation Instance ID for operations 32-63 is the first byte of the argument. Duplicate Detection suuport strips that byte.
- The Instance ID is not subject to Basic Encoding Rules (BER).
- The invoker user assigns the Operation Instance Identifier to the operation at the time of requesting the invoke service. The Operation Value should be in the range of operation values with duplication detection support, i.e. 32 to 63.
- It’s the responsibility of the user to choose Operation Instance Identifier in a way that uniqely and unambiguously identifies the operation.
- From the invoker’s perspective, assumption is that two operations with the same operation Instance Identifier are totally identical which means they produce exact same results.

- Operation Instance Identifier uniqely specifies a non-idempotent operation and multiple invocations of such an operation will eventually result in the same outcome because the duplicate instances are identified and the operation is not performed more than once.
- From the performer's perspective, assumption is that two operations with the same Operation Instance Identifier should be executed once and once only.
- If requested, the degree of duplication checked by Duplicate Operation Detection Support Functional Unit on the performer's side (i.e. the total number of outstanding Operation Instance Identifier kept) can be communicated with the invoker to synchronize the invocations.
- User of Duplicate Operation Detection Support is responsible to behave based on the performer profile and its limitations in this regard. This behavior is defined based on the desired semantic of the operation which is to be implemented.
- On the performer side, when an Operation Instance Identifier is received, a previous Operation Instance Identifier whose distance to this latest one is greater than or equal to half of the wrap-around range of the Operation Instance Identifier number is expired, i.e. for an 8-bit Operation Instance Identifier, the distance of 128 causes an old Operation Instance Identifier to expire.
- It's the responsibility of the invoker user to use consecutive Operation Instance Identifier numbers, or when it skips some Operation Instance Identifiers, it should remember that if there is an smaller Operation Instance Identifier on performer side with the distance explained above, it will be expired.

5 EMSD PROCEDURE FOR OPERATIONS

The following sections shows the general procedures to be used in the implementation of the EMSD Message Transfer Server (MTS) and the EMSD User Agent (UA), with the option for 3-Way or 2-Way handshakes on operations which support them. These procedures do not constitute complete behavior specifications for implementations. The following sections contain information helpful to implementors.

The MTS and the UA are event-driven. Each waits for any of the possible event types, and, upon receiving an event, processes it. After processing the event, the next event is waited upon.

5.1 MTS Behavior

The MTS is event-driven.

If it received an event from ESROS, then it could be any of the following types:

- Message submit indication;
- Message submit confirm and failure indication;
- Result and Error indication for a deliver operation;
- DeliveryVerify indication;
- Result and Error indication for a submissionVerify operation;

- Result and Error indication for a submissionControl operation;
- DeliveryControl indication.

For an ESROS event responsibility is passed to the MTS performer (Section 5.1.1).

If the MTS received an event:

- for message delivery, from the RFC-822 mailer;
- requesting submission controls upon the UA, or;
- indicating an elapsed timer (meaning that it's time to re-attempt a message delivery)

then responsibility is passed to the MTS invoker (Section 5.1.5).

5.1.1 MTS Performer

The MTS performer is responsible for processing the following operations, received from ESROS:

- Message-submission
- Delivery-control
- Delivery-verify

The MTS performer should first make sure that it has received an INVOKE.indication. Any other type of primitive shouldn't be occurring at this point, and should be ignored.

If there's something wrong with the PDU or operation data, the MTS performer should send back an error to the proper invoker:

1. Send an ESROS Error Request, then go wait for a response (either a confirmation or a failure indication). The response is sent back on the same SAP type on which the event occurred.
2. Keep track of the type of request that was issued.

If there isn't anything wrong with the PDU or operation data, then the MTS performer has received a valid event from ESROS. This could be any of the defined Submission and Delivery Protocol operations.

5.1.2 Message-submission

1. The Message-submission operation first checks to see which SAP this Submit Request came in on.
2. The request could have arrived as 2-Way SAP (see #3) or a 3-Way SAP (see #7).

3. If the event arrived on the 2-Way SAP, consider this a protocol violation and ignore it.
4. Wait for a response to the request. The response could be either an ERROR.confirm (see #5) or a FAILURE.indication (see #6).
5. The ERROR.request has been confirmed. The UA knows that the submitted message wasn't sent. Since there was an error, there is nothing more to do, so return.
6. If the result to the ErrorRequest is a Failure.indication, it can be assumed that either the UA has received nothing (the ERROR.request PDU was lost), which means failure for the UA; or that the 3-Way acknowledgment was lost, which means that the UA has in fact received the ERROR.request PDU and knows about the delivery failure. Either way, the message can be ignored. There is nothing more to do, so return.
7. If the event was received on the 3-Way SAP, then this is the correct SAP on which to receive a Submit Request. Send back a Result Request and keep track of the primitive which was issued.
8. Now wait for a response to our request. The response will be either a Result.confirm (see #9) or a Failure.indication (see #13).
9. The RESULT.request has been confirmed.
10. Submit the message to the RFC-822 mailer.
11. Attempt, a number of times, to send the submitted message via the RFC-822 mailer. If the send was successful, then return.
12. If, after the maximum number of retries, the message was not able to be sent, consider it a failure. Since the UA assumption has been that submission was successful, but now it has not been sent, a brand new message, a Non-Delivery message, must be generated and delivered to the UA. When this is completed, then return.
13. A FAILURE.indication has occurred due to the previously issued RESULT.request.
14. A Submission Verification is issued to the UA to see if the RESULT.request was received. There are three possible results from sending the submission verification to the UA: Fail (see #15), Send Message (see #16) or Drop Message (see #20).
15. Fail – The Submission-verify request didn't reach the UA, or the Submission Verify response didn't get back. Ignore the message and return.
16. The Submission Verify operation succeeded, meaning that the UA received the request, and responded with a message stating that it wants the message to be sent.
17. Attempt, a number of times, to send the submitted message via the RFC-822 mailer.
18. If the message was submitted to the RFC-822 mailer successfully, then return. If, after the maximum number of retries, the message was not able to send the message, consider it a failure.
19. The UA already assumes that the Message-submission was successful. Now since the submitted message has not been sent, a brand new message, a Non-Delivery message, must be generated and delivered to the UA. After this is accomplished, then return.
20. The UA responded with a message stating that the message should be dropped. This may occur if the UA never received the result from the MTS, meaning that it never received the Message Id, and had to therefore inform the user that the message couldn't be submitted. This may also occur if the UA doesn't have the record of the message being verified. It can be because the message record has been aged and expired, or because the EMSD-UA has not been able to keep the record of the received message because of storage or memory limitations. There is nothing to do, so return.

5.1.3 Delivery-control

This operation can be processed immediately. After it is processed, the appropriate result is returned.

5.1.4 Delivery-verify

This operation occurs when the UA doesn't think that the MTS has received the RESULT.indication from a previously delivered message. The UA wants to make sure that the MTS knows it has been delivered. The MTS will determine what it knows of the specified message, and send back a result. This can be processed immediately, as it doesn't need to deal with duplicate detection.

5.1.5 MTS Invoker

The MTS invoker is responsible for processing the following operations, received from ESROS:

- Message-delivery
- Submission-control
- Submission-verify

Submission-control

Process the Submission Control request.

Message-delivery

1. Check the User Agent's profile to determine the SAP.
2. Set the SAP to 3-Way.
3. Issue the INVOKE.request on the appropriate SAP, with duplication detection enabled. Since a local error is possible on issuing the INVOKE.request, a retry counter is needed.
4. There are three possible events possible in result to the INVOKE.request: an ERROR.indication (see #5), a RESULT.indication (see #9) or a FAILURE.indication (see #10).
5. An ERROR.indication was received, which means that the UA can't accept the message right now.
6. If the reason was one of a transient nature, wait for a while and then send the Deliver Request again.
7. If the reason was one of a permanent nature, send back a non-delivery report to the originator.
8. Since the error was one of a permanent nature, then the MTS must send back a non-delivery report, then log the unsuccessful delivery with error from UA and return.

9. A RESULT.Indication was returned, which means that the Delivery was successful. Send a delivery report to the originator if one was requested and log successful delivery and return.

   If the UA profile indicated that Complete mode was to be used, keep track of the fact that this message has been successfully delivered (as far as the MTS is concerned), so that if the UA sends us a Delivery Verify operation, we know that we consider the message to be delivered.

10. A FAILURE.indication was returned, which means there was a problem getting the Deliver Request to the UA, or in getting the response back from the UA. In any case, a response was never received, so the request timed out. Wait for a while, and then send the Deliver Request again.

    As long as a FAILURE.indication is returned and the number of retries has not been exceeded, keep trying to verify the delivery.

Submission-verify

The Submission-verify operation is always issued on the 2-Way SAP. The response is awaited. If a response doesn't come, the request is queued and attempted again later.

1. Issue the INVOKE.request on the 2-Way SAP, with duplication detection disabled. Since a local error on issuing the invoke request is possible, a retry counter is needed.
2. An INVOKE.Request has been issued and a response has been received. The response will be either a a RESULT.indication (see #3) or a FAILURE.indication (see #4). There are no defined errors to a Submission Verify operation, so an ERROR.indication should not be occurring here.
3. A RESULT.indication was received. Either ResponseSendMessage or ResponseDropMessage, as specified in the PDU, will be returned.
4. A FAILURE.indication was received, which means that there was a problem getting the Submission Verify Request to the UA, or in getting the response back from the UA. In any case, the response was never received, so the request timed out. Wait for a while, and then another attempt to send the Submission Verify request is needed.

Non-Delivery Report

Issue an INVOKE.request containing a Submit operation with a content type of Non- Delivery Report, to the UA. This operation is always issued on the 2-Way SAP. The response is awaited. If a response doesn't come, the request is queued and attempted again later.

1. Create a Submit operation.
2. Issue the INVOKE.request on the 2-Way SAP, with duplication detection enabled. Since a local error on issuing the invoke request is possible, a retry counter for is needed.
3. A response to the INVOKE.Request has been received. The response will be either a RESULT.indication (see #5), ERROR.indication (see #4), or a FAILURE indication (see #7).
4. An ERROR.indication was received, which means that the UA doesn't know what to do with our non-delivery report. That's the UAs problem, so just do nothing and return.

5. A RESULT.indication was received, which means we delivered a successful non-delivery report.
6. The result is logged. Nothing more is needed, so return.
7. A FAILURE.indication was received, which means there was a problem getting the Submit Request to the UA, or in getting the response back from the UA. In any case, the response was never, so the request timed out. Wait for a while, and then send the Submission Verify request again.

5.2 UA Behavior

The User Agent is event-driven.

If it received an event from ESROS, then it could be any of the following types:

- Message deliver indication;
- Message deliver confirm and failure indication;
- Result and Error indication for a submit operation;
- Submission verify indication;
- Result and Error indication for a delivery verify operation;
- Result and Error indication for a delivery control operation;
- Submission control indication.

For an ESROS event responsibility is passed to the UA performer (Section 5.2.1).

IF the UA received an event indicating that there's a message from the user, for submission, then responsibility is passed to the UA invoker (Section 5.2.2).

5.2.1 UA Performer

The performer on the UA side is responsible for processing the following operations:

- Message Delivery
- Submission Verification
- Submission Control

Message-delivery

1. A Message-delivery request is received.
2. Check for the correctness of the PDU. If the PDU is bad the see #3. If the PDU is good then see #8.
3. Send an ESROS ERROR.request. If the request arrived on a 3-Way SAP, use a 3-Way SAP for the result. If the request arrived on a 2-Way SAP, use a 2-Way SAP for the result. Keep track of the type of request that was issued.
4. Wait for the ESROS event. The result could be an ERROR.confirm (see #5) or a FAILURE.indication (see #7).
5. The ESROS event was an ERROR.confirm
6. Log the message as the Non-Delivery was confirmed by the MTS and return.
7. If the ESROS event was a FAILURE.indication, that means one of two things has occurred:
   A. The MTS has received nothing (the ERROR.request PDU was lost), which means that the MTS doesn't know that the message delivery has been rejected. In this case, the MTS will eventually time out, and retransmit the message delivery request.
   B. The 3-Way acknowledgment was lost, which means that the MTS has in fact received the ERROR.request PDU and knows about the delivery failure.

   Either way, the message can now be ignored.
8. Send an ESROS RESULT.request. If the request arrived on a 3-Way SAP, use a 3-Way SAP for the result. If the request arrived on a 2-Way SAP, use a 2-Way SAP for the result. Keep track of the type of request that was issued.
9. Wait for the ESROS event. The result could be an RESULT.confirm (see #10) or a FAILURE.indication (see #13).
10. If the event is a RESULT.confirm, then the delivered message can now be given to the user.
11. Deliver the message to the user.
12. Log the message as Message Delivery Known to MTS.
13. If the event is a FAILURE.indication, then, if the delivery was on a 3-Way SAP, a Delivery Verification request to the MTS can be issued to see if the MTS actually got the RSULT.request. If the delivery was on a 2-Way SAP, then the message will delivered to the user and if the MTS has not received the RESULT.request, it will retransmit it later and the duplicate will be ignored.
14. Deliver the message to the user. Since a FAILRUE.indication was received in response to a RESULT.requst, it means that possible, the MTS didn't receive the RESULT.request. The MTS could now time out, and send another copy of the same message. Save the message for duplication detection.
15. Log the fact that the message was delivered, but that the MTS might not be aware of it.
16. If the UA supports Delivery Verification, and the Delivery Request was sent on the 3-Way SAP, then see #17. If either of these conditions are not true, then return.
17. Send a Delivery-verify request to see if the MTS got the RESULT.request.

    There are three possible results from sending the delivery verification to the MTS: Fail (see #18), ResponseNonDelivery (see #20) or ResponseDelivery (see #23).

18. Fail – Delivery Verify request didn't reach the MTS, or the Delivery Verify response didn't get back to the UA.

19. Log this as delivering the message to the user, but the MTS having possibly sent a Non-Delivery report to the originator even though the UA did actually deliver the message to the user. Then return.

20. ResponseNonDelivery – Verify Response indicates that the MTS now knows (because of the Delivery Verify operation that the message has been delivered to the user, but had not received our RESULT.request nor a Delivery Verify operation in a timely manner, and had already sent out a Non-Delivery report to the originator.

21. The MTS had not received, from the UA, in a timely manner, a RESULT.indication indicating that the message had been delivered to the user. The MTS has already sent a Non-Delivery report to the originator. The UA must let the user know about this. Log the message as delivered to the user, but a Non-Delivery sent to the originator.

22. Since the UA received a response to the Verify operation, it knows that the MTS knows about this message delivery, so the UA also knows that it won't be receiving a duplicate of it. The UA can now remove this message's Message Id from the list of possible duplicates.

23. ResponseDelivery – Verify Response received from MTS.

24. This means that the MTS knows (either because the MTS had received the RESULT.request that was sent by the UA or because the MTS has now received the UAs Delivery-verification message, informing that the UA received the message for delivery to the user. The MTS is (or was) able to send a Delivery report to the originator if one was requested. Log it as such.

25. Since the UA received a response to the Verify operation, it knows that the MTS knows about this message delivery, so the UA also knows that it won't be receiving a duplicate of it. The UA can now remove this message's Message Id from the list of possible duplicates and return.

Submission-verify

Process the Submission-verify request and return.

Submission-control

This operation can be processed immediately. After it is processed, the appropriate result is returned.

5.2.2 UA Invoker

The invoker on the UA side is responsible for processing the following operations:

- Message-submission
- Delivery-control
- Delivery-verify

Message-submission

General procedures for UA's Message-submission mirror that of MTS's Message-delivery.

Delivery-control

1. Issue the INVOKE.request on the 3-Way SAP, with duplication detection enabled. Since the UA can get a local error on issuing the invoke request, a retry counter is needed.

   If we got a local failure in issuing the Invoke Request, wait a while and then try again (up to the limit of the maximum number of retries).

2. The UA has issued an INVOKE.Request. Wait for a response from ESROS. The response will be either a RESULT.indication (see #5), ERROR.indication (see #3), or FAILURE.indication (see #7).
3. A ERROR.indicaiton was received, meaning that the MTS told says that it cannot accept the message.
4. Log the MTS rejection and return
5. A RESULT.indication was received, which means that the Submission was successful.
6. Log successful submission and return.
7. a FAILURE.indication was received, meaning that there was a problem getting the Submit Request to the MTS, or in getting the response back from the MTS. In any case, the UA never received the response, so the request timed out. Wait for a while, and then send the Submit Request again.
8. The UA has exceeded the maximum number of retries. Let the user know, log the failure and return.

Delivery-verify

General procedures for UA's Delivery-verify mirror that of MTS's Submission-verify.

6 EMSD FORMAT STANDARDS

6.1 Format Standard Overview

EMSD Format Standard (EMSD-FS) is a non-textual form of compact encoding of Internet mail (RFC-822) messages which facilitates efficient transfer of messages. EMSD-FS is used in conjunction with the EMSD-P but is not a general replacement for RFC-822. EMSD-FS defines a method of representation of short interpersonal message. It defines the "Content" encoding (Header + Body). Although EMSD-FS contains end-to-end information its scope is purely point-to-point.

The "Efficient InterPersonal Message Format Standard" is defined in this section. This standard is primarily intended for communication among people.

The EMSD Format Standard is designed to be fully consistent with RFC-822 [**?**]. In many ways EMSD-FS can be considered to be an efficiency oriented encoder and decoder. Through use of EMSD-FS an RFC-822 message is converted to a more compact binary encoding. This more compact message is then transfered between an EMSD-SA and EMSD-UA. The compact message (represented in EMSD-FS) may then be converted back to RFC-822 intact.

For messages that are originated (submitted) with EMSD protocol, certain fields (e.g., addresses, message-id) can have special forms that are specialized and produce more compact EMSD-FS encoding. These special forms are legitimate values of RFC-822 messages.

This specification expresses information objects using ASN.1 [X.208]. Encoding of ASN.1 shall be based on Basic Encoding Rules (BER) [**?**]. Future revisions of this specification will use Packed Encoding Rules (PER) [**?**].

The convention of (O) ”OPTIONAL”, (D) ”DEFAULT”, (C) ”CONDITIONAL” and (M) ”MANDATORY” which express requirements for presence of information is used in this section.

6.2 Interpersonal Messages

An interpersonal message (IPM) consists of a heading and a body.

```
IPM ::=   SEQUENCE

{

  heading       Heading,

  body          Body OPTIONAL

};
```

6.2.1 Heading fields

The fields that may appear in the Heading of an IPM are defined and described below.

```
Heading ::= SEQUENCE
{
  -- Address of the sending agent (person, program, machine) of
  -- this message. This field is mandatory if the sender
  -- is different than the originator.
  sender                      [0]     EMSDORAddress OPTIONAL,

  -- Address of the originator of the message
  -- (not necessarily the sender)
  originator                          EMSDORAddress,

  -- List of recipients and flags associated with each.
```

```
recipient-data                        SEQUENCE SIZE (1..ub-recipients)
                                      OF PerRecipientFields,

-- Flags applying to this entire message
per-message-flags           [1]     IMPLICIT BIT STRING

 {
 -- Priority values
 -- At most one of "non-urgent" and "urgent" may be specified
 -- concurrently.  If neither is specified, then a Priority
 -- level of "normal" is assumed.
 priority-non-urgent             (0),
 priority-urgent                 (1),

 -- Importance values
 -- At most one of "low" and "high" may be specified
 -- concurrently.  If neither is specified, then an
 -- Importance level of "normal" is  assumed.
 importance-low                  (2),
 importance-high                 (3),

 -- Indication of whether this message has been
    automatically forwarded
 auto-forwarded                  (4)
 } OPTIONAL,

-- User-specified recipient who is to receive replies
   to this message.
reply-to                    [2]     IMPLICIT SEQUENCE SIZE
                                    (1..ub-reply-to)
                                   OF EMSDORAddress OPTIONAL,

-- Identifier of a previous message, for which this message
-- is a reply
replied-to-IPM                       EMSDMessageId OPTIONAL,

-- Subject of the message.
subject                     [3]     IMPLICIT AsciiPrintableString
                                    (SIZE (0..ub-subject-field))
                                              OPTIONAL,

-- RFC-822 header fields not explicitly provided for in
-- this Heading. For messages incoming from the external
-- world (i.e. in RFC-822 format), the Message-Id: field
-- need not go here, as it is placed in the
-- Envelope's EMSDMessageId (message-id) field.
extensions        [4]  IMPLICIT  SEQUENCE
                       (SIZE (0..ub-header-extensions))
                       OF  IPMSExtension OPTIONAL,

-- MIME Version (if other than 1.0)
```

```
  mime-version            [5]     IMPLICIT AsciiPrintableString
                                  (SIZE (0..ub-mime-version-length))
                                                OPTIONAL,

  -- Top-level MIME Content Type
  mime-content-type       [6]     IMPLICIT AsciiPrintableString
                                  (SIZE (0..
                                   ub-mime-content-type-length))
                                            OPTIONAL,

  -- MIME Content Id
  mime-content-id         [7]     IMPLICIT AsciiPrintableString
                                  (SIZE (0..
                                   ub-mime-content-id-length))
                                            OPTIONAL,

  -- MIME Content Description
  mime-content-description [8]    IMPLICIT AsciiPrintableString
                                  (SIZE (0..ub-mime-content-
                                  description-length))
                                            OPTIONAL,
  -- Top-level MIME Content Type
  mime-content-transfer-encoding
                           [9]     IMPLICIT AsciiPrintableString
                                   (SIZE (0..ub-mime-content-
                                   transfer-encoding))
                                            OPTIONAL
};
```

Some fields have components and thus are composite, rather than indivisible. A field component is called a sub-field.

Sender

This field is mandatory if the sender is different from the originator.

Originator

The Originator heading field (O) identifies the IPM's originator.

Recipient-data

```
PerRecipientFields ::= SEQUENCE
{
  recipient-address                             EMSDORAddress,
```

```
  per-recipient-flags                         BIT STRING

  {
  -- Recipient Types.
  -- At most one of "copy" and "blind-copy" may be
  -- specified concurrently for a single recipient.  If
  -- neither is specified, than the recipient
  -- is assumed to be a "primary" recipient.
  recipient-type-copy                            (0),
  recipient-type-blind-copy                      (1),

  -- Notification Request Types.
  -- Only one of "rn" and "nrn" may be specified
  -- concurrently, \\x110011 for a single recipient.
  -- "rn" implies "nrn" in addition.
  notification-request-rn                        (2),
  notification-request-nrn                       (3),

  notification-request-ipm-return                (4),

  -- Report Request Types
  -- At most one of these should be set for a
  -- particular recipient. "delivery" implies "non-delivery"
  -- in addition.
  report-request-non-delivery                    (5),
  report-request-delivery                        (6),

  -- Originator-to-Recipient request for a reply.
  reply-requested                                (7)
  } DEFAULT { report-request-non-delivery }

};
```

recipient-address

The Primary Recipients heading field identifies the zero or more users who are the ”primary recipients” of the IPM. The primary recipients might be those users who are expected to act upon the IPM.

per-recipient-flags

The Copy Recipients heading field identifies the zero or more users who are the ”copy recipients” of the IPM. The copy recipients might be those users to whom the IPM is conveyed for information.

recipient-type-copy

This field is set if the recipient is on the Carbon Copy (CC) list.

recipient-type-blind-copy

This field is set if the recipient is on the Blind Carbon Copy (BCC) list.

The Blind Copy Recipients heading field (C) identifies zero or more users who are the intended blind copy recipients of the IPM.

The phrase "copy recipients" above has the same meaning as in "Copy Recipients" from Section 6.2.1 . A blind copy recipient is one whose role as such is disclosed to neither primary nor copy recipients.

In the instance of an IPM intended for a blind copy recipient, this conditional field shall be present and identify that user. Whether it shall also identify the other blind copy recipients is a local matter. In the instance of the IPM intended for a primary or copy recipient, the field shall be absent.

notification-request-rn

A receipt notification (rn) reports its originator's receipt, or his expected and arranged future receipt, of an IPM.

notification-request-nrn

A non-receipt notification (nrn) reports its originator's failure to receive, to accept, or his delay in receiving, an IPM.

notification-request-ipm-return

When this field is set, the contents of the message are returned along with the notification.

report-request-non-delivery

The report request enables the MTS to acknowledge to the MTS-user one or more outcomes of a previous invocation of the message-submission or probe-submission abstract-operations.

A report is returned only in case of non-delivery.

report-request-delivery

For the message-submission, report-delivery indicates the delivery or non-delivery of the submitted message to one or more recipients. For the probe-submission, the report- delivery indicates whether or not a message could be delivered if the message were to be submitted.

reply-requested

When set this field indicates that the originator requests that a recipient send a message in reply to the message which carries the request.

per-message-Flags

Priority

The Priority field (default is normal) identifies the priority that the authorizing users attach to the IPM. It may assume any one of the following values: urgent, normal, or non-urgent.

At most one of either ”non-urgent” or ”urgent” may be specified concurrently. If neither is specified, then a Priority level of ”normal” is assumed.

Importance

The Importance heading field (default normal) identifies the importance that the authorizing users attach to the IPM. It may assume any one of the following values: low, normal, or high.

At most one of either ”low” or ”high” may be specified concurrently. If neither is specified, then a Importance level of ”normal” is assumed.

The values above are not defined by this specification; they are given meaning by users.

auto-forwarded

The Auto-forwarded heading field (default is false) indicates whether the IPM is the result of auto-forwarding. It is a Boolean value.

reply-to

User-specified recipient or recipients who are to receive replies to this message.

replied-to IPM

The Replied-to IPM heading field (C) identifies the IPM to which the present IPM is a reply. It comprises an IPM identifier.

This conditional field shall be present if, and only if, the IPM is a reply.

Note - In the context of forwarding, care should be taken to distinguish between the forwarding IPM and the forwarded IPM. This field should identify whichever of these two IPMs to which the reply responds.

subject

The Subject heading field (O) identifies the subject of the IPM. It corresponds to the "Subject:" field of RFC-822.

extensions

The Extensions heading field [D no extensions (i.e. members)] conveys information accommodated by no other heading field. It comprises a Set of zero or more IPMS extensions, each conveying one such information item.

```
IPMSExtension ::= SEQUENCE
{
    x-header-label                      AsciiPrintableString,
    x-header-value                      AsciiPrintableString
};
```

6.2.2 Body part types

The types of body parts that may appear in the Body of an IPM are structured using the MIME specification.

```
Body ::= SEQUENCE
{
  compression-method          [0]     IMPLICIT CompressionMethod
                                               OPTIONAL,
  -- If compression method is not specified,
  -- "no-compression" is implied.

  message-body                        OCTET STRING
  -- See MIME for structure of the Body.
  -- If a compression method is specified, the entire text containing
  -- the Content-Type: element followed by the RFC-822 body are
  -- compressed using the specified method, and placed herein.
};

CompressionMethod ::= INTEGER
{
  -- Compression Methods numbered 0 to 63 are reserved for
  -- assignment within this and associated specifications.
  no-compression                  (0),
```

```
  lempel-ziv                      (1)

  -- Compression Methods numbered between 64 and 127 may be
  --  used on a bilaterally-agreed basis between peers.
} (0..127)
```

7 ACKNOWLEDGMENTS

In the context of Limited Size Messaging (LSM) over CDPD and pACT over Narrowband PCS, AT&T Wireless Services (AWS), funded work which was relevant to the development of the EMSD protocols.

8 SECURITY CONSIDERATIONS

This protocol supports simple authentication of the originator's address by the EMSD-SA and simple authentication of EMSD-SA by EMSD-UA.

Mainstream Internet mail security mechanisms can be used in conjunction with the EMSD protocol.

9 AUTHOR'S ADDRESS

Mohsen Banan
Neda Communications, Inc.
17005 SE 31st Place
Bellevue, WA 98008
email: http://mohsen.banan.1.byname.net/ContactMe

A EMSD-P ASN.1 MODULE

This section compiles in one place the complete ASN.1 Module for EM Submission and Delivery Protocol.

```
EMSD-SubmissionAndDeliveryProtocol DEFINITIONS ::=

BEGIN

EXPORTS EMSDORAddress, AsciiPrintableString, ContentType,
DateTime, EMSDMessageId, EMSDORAddress, ProtocolVersionNumber;

-- Upper bounds
```

```
ub-recipients  INTEGER ::= 256;
-- also defined in EMSD-InterpersonalMessaging1995
ub-reply-to INTEGER ::= 256;
-- also defined in EMSD-InterpersonalMessaging1995
ub-subject-field INTEGER ::= 128;
-- also defined in EMSD-InterpersonalMessaging1995
ub-password-length INTEGER ::= 16;
ub-content-length INTEGER ::= 65535;
-- also defined in EMSD-Probe
ub-content-types INTEGER ::= 128;
ub-message-id-length INTEGER ::= 127;
ub-total-number-of-segments INTEGER ::= 32;
ub-header-extensions INTEGER ::= 64;
-- also defined in EMSD-InterpersonalMessaging1995
ub-emsd-name-length INTEGER ::= 64;
ub-emsd-address-length INTEGER ::= 20;
ub-rfc822-name-length INTEGER ::= 127;
ub-mime-version-length INTEGER ::= 8;
-- also defined in EMSD-InterpersonalMessaging1995
ub-mime-content-type-length INTEGER ::= 127;
-- also defined in EMSD-InterpersonalMessaging1995
ub-mime-content-id-length INTEGER ::= 127;
-- also defined in EMSD-InterpersonalMessaging1995
ub-mime-content-description-length INTEGER ::= 127;
-- also defined in EMSD-InterpersonalMessaging1995
ub-mime-content-transfer-encoding INTEGER ::= 127;
-- also defined in EMSD-InterpersonalMessaging1995
ub-local-message-nu INTEGER ::= 4096;

----------------------
-- SUBMIT Operation --
----------------------

submit ES-OPERATION

    ARGUMENT SubmitArgument
    RESULT SubmitResult
    ERRORS
    {
        submissionControlViolated,
        securityError,
        resourceError,
        protocolViolation,
        messageError
    } ::= 33;

SubmitArgument ::= SEQUENCE
{
  -- Security features
  security           [0]    IMPLICIT SecurityElement
```

```
                            OPTIONAL,

  -- Segmentation features for efficient transport
  segment-info                  SegmentInfo OPTIONAL,

  -- Content type of the message
  content-type                            ContentType,

  --
  -- THE CONTENT --
  --

  -- The submission content
  content                       ANY DEFINED BY content-type

};

SubmitResult ::= SEQUENCE

{

  -- Permanent identifier for this message.
  -- Also contains the message submission time.
  -- See comment regarding assignment of message
  -- identifiers, at the definition of EMSDLocalMessageId.
  message-id                        EMSDLocalMessageId
    };

--------------------------------
-- Delivery Control Operation --
--------------------------------

deliveryControl ES-OPERATION
    ARGUMENT DeliveryControlArgument
    RESULT DeliveryControlResult
    ERRORS
    {
        securityError,
        resourceError,
        protocolViolation
    } ::= 2;

DeliveryControlArgument ::= SEQUENCE
{
  -- Request an addition of or removal of a set of restrictions
  restrict             [0]     IMPLICIT Restrict DEFAULT update,

  -- Which operations are to be placed in the restriction set
  permissible-operations  [1]     IMPLICIT Operations OPTIONAL,

  -- What maximum content length should be allowed
```

```
  permissible-max-content-length
                        [2]     IMPLICIT INTEGER
                                (0..ub-content-length) OPTIONAL,

  -- What is the lowest priority message which may be delivered
  permissible-lowest-priority
                        [3]     IMPLICIT ENUMERATED
                                {
                                   non-urgent     (0),
                                   normal         (1),
                                   urgent         (2)
                                } OPTIONAL,

  -- Security features
  security                [4]     IMPLICIT SecurityElement
                                              OPTIONAL,

  -- User Feature selection
  user-features           [5]     IMPLICIT OCTET STRING OPTIONAL
};

DeliveryControlResult ::= SEQUENCE
{
  -- Operation types queued at the EMSD-SA due to existing
  -- restrictions.
  waiting-operations    [0]   IMPLICIT Operations DEFAULT { },


  -- Types of messages queued at the EMSD-SA due to
  -- existing restrictions
  waiting-messages      [1]   IMPLICIT WaitingMessages DEFAULT { },

  -- Content Types of messages queued at the EMSD-SA
  waiting-content-types   SEQUENCE SIZE (0..ub-content-types) OF
                                    ContentType DEFAULT { }
};

Restrict ::= ENUMERATED
{
    update                                      (1),
    remove                                      (2)
};

Operations ::= BIT STRING
{
    submission                                  (0),
    delivery                                    (1)
};


WaitingMessages ::= BIT STRING
```

```
{
    long-content                                (0),
    low-priority                                (1)
};

-- Delivery Verify Operation

deliveryVerify ES-OPERATION

    ARGUMENT DeliveryVerifyArgument
    RESULT DeliveryVerifyResult
    ERRORS
    {
        verifyError,
        resourceError,
        protocolViolation
    } ::= 5;

DeliveryVerifyArgument ::= SEQUENCE
{
  -- Identifier of this message. This is the same identifier that
  -- was provided to the originator in the Submission Result.
  -- See comment regarding assignment of message identifiers,
  -- at the definition of EMSDMessageId.
  message-id                                      EMSDMessageId
};

DeliveryVerifyResult ::= SEQUENCE
{
                         status  DeliveryStatus
};

 DeliveryStatus  ::= ENUMERATED
{
        no-report-is-sent-out                   (1),
        delivery-report-is-sent-out             (2),
        non-delivery-report-is-sent-out         (3)
};

-----------------------
-- DELIVER Operation --
-----------------------

deliver ES-OPERATION
    ARGUMENT DeliverArgument
    RESULT NULL
    ERRORS
    {
        deliveryControlViolated,
        securityError,
        resourceError,
```

```
        protocolViolation,
        messageError
    } ::= 35;

DeliverArgument ::= SEQUENCE
{
  -- Identifier of this message. This is the same identifier that
  -- was provided to the originator in the Submission Result.
  -- See comment regarding assignment of message identifiers,
  -- at the definition of EMSDMessageId.
  message-id                                      EMSDMessageId,

  -- Time the message was delivered to the recipient by EMSD-SA
  message-delivery-time                           DateTime,

  -- Time EMSD-SA originally took responsibility for processing
  -- of this message. This field shall be omitted if the message-id
  -- contains an EMSDLocalMessageId, because that field contains
  -- the submission time within it.
  message-submission-time [0]     IMPLICIT   DateTime OPTIONAL,

  -- Security features
  security                [1]     IMPLICIT   SecurityElement OPTIONAL,

  -- SegContentTypementation features for efficient transport
  segment-info                               SegmentInfo OPTIONAL,

  -- The type of the content
  content-type                               ContentType,

  --
  -- THE CONTENT --
  --

  -- The submitted (and now being delivered) content
  content                       ANY DEFINED BY content-type
};

-- Submission Control Operation

submissionControl ES-OPERATION
    ARGUMENT SubmissionControlArgument
    RESULT SubmissionControlResult
    ERRORS
    {
        securityError,
        resourceError,
        protocolViolation
    } ::= 4;

SubmissionControlArgument ::= SEQUENCE
```

```
{
  -- Request an addition of or removal of a set of restrictions
  restrict               [0]     IMPLICIT Restrict DEFAULT update,

  -- Which operations are to be placed in the restriction set
  permissible-operations  [1]     IMPLICIT Operations OPTIONAL,

  -- What maximum content length should be allowed
  permissible-max-content-length
                          [2]     IMPLICIT INTEGER
                                  (0..ub-content-length) OPTIONAL,

  -- Security features
  security                [3]     IMPLICIT SecurityElement
                                                  OPTIONAL
};

SubmissionControlResult ::= SEQUENCE
{
  -- Operation types queued at the EMSD-SA due to existing
  -- restrictions.
  waiting-operations    [0]   IMPLICIT Operations DEFAULT { }

};

----------------------------------
-- Submission Verify Operation --
----------------------------------

submissionVerify  ES-OPERATION

    ARGUMENT SubmissionVerifyArgument
    RESULT SubmissionVerifyResult
    ERRORS
    {
        submissionVerifyError,
        resourceError,
        protocolViolation
    } ::= 6;

SubmissionVerifyArgument ::= SEQUENCE
  -- Identifier of this message. This is the same identifier that
  -- was provided to the originator in the Submission Result.
  -- See comment regarding assignment of message identifiers,
  -- at the definition of EMSDMessageId.
  {
     message-id                        EMSDMessageId
  };

SubmissionVerifyResult ::= SEQUENCE
    {
```

```
        status  SubmissionStatus
    };

SubmissionStatus::= ENUMERATED
{
        send-message            (1),
        drop-message            (2)
};

-- GetConfiguration Operation
-- To be fully defined later. This will possibly include,
-- but not be limited to:
--      get-local-time-zone
--      get-protocol-version
--      etc.

getConfiguration ES-OPERATION

        ARGUMENT NULL
        RESULT NULL
        ERRORS
        {
            resourceError,
            protocolViolation
        } ::= 7;

-- SetConfiguration Operation
-- To be fully defined later.

setConfiguration ES-OPERATION

        ARGUMENT NULL
        RESULT NULL
        ERRORS
        {
            resourceError,
            protocolViolation
        } ::= 8;

-- Security --

SecurityElement ::= SEQUENCE

{
  credentials                   Credentials,
  contentIntegrityCheck         ContentIntegrityCheck OPTIONAL
};

Credentials ::= CHOICE
{
  simple                          [0]   IMPLICIT SimpleCredentials
```

```
  -- Strong Credentials are for future study
  -- strong                       [1]   IMPLICIT StrongCredentials
  -- externalProcedure            [2]   EXTERNAL
};

SimpleCredentials ::= SEQUENCE

{
  eMSDAddress                         EMSDAddress OPTIONAL,
  password                    [0]     IMPLICIT OCTET STRING
                              (SIZE (0..ub-password-length)) OPTIONAL
};

-- StrongCredentials ::= NULL
-- for now.

-- ContentIntegrityCheck is a 16-bit checksum of content
ContentIntegrityCheck ::= INTEGER (0..65535);

SegmentInfo ::= CHOICE

{
  first           [APPLICATION 2]         IMPLICIT FirstSegment,
  other           [APPLICATION 3]         IMPLICIT OtherSegment
};

FirstSegment ::= SEQUENCE

{
  sequence-id                             INTEGER,
  number-of-segments                      INTEGER
  -- number-of-segments must not exceed ub-total-number-of-segments

};

OtherSegment ::= SEQUENCE
{
  sequence-id                             INTEGER,
  segment-number                          INTEGER
};

-----------
-- Errors --
------------

protocolVersionNotRecognized  ERROR PARAMETER NULL ::= 1;

submissionControlViolated  ERROR PARAMETER NULL ::= 2;

messageIdentifierInvalid  ERROR PARAMETER NULL ::= 3;
```

```
securityError ERROR PARAMETER security-problem SecurityProblem ::= 4;

deliveryControlViolated   ERROR PARAMETER NULL ::= 5;

resourceError  ERROR PARAMETER NULL ::= 6;

protocolViolation  ERROR PARAMETER NULL ::= 7;

messageError  ERROR PARAMETER NULL ::= 8;

SecurityProblem ::= INTEGER (0..127);


--
-- EXPORTED Definitions (for use by associated specifications) --
--

ContentType ::=  INTEGER
{
  -- Content type 0 is reserved and shall never be transmitted.
  reserved                                (0),
  -- Content types between 1 and 31 (inclusive) are for
  -- internal-use only
  probe                                   (1), -- reserved
  delivery-report                         (2), -- reserved

  -- Content types between 32 and 63 (inclusive) are for
  -- message types  defined within this specifications.
  emsd-interpersonal-messaging-1995       (32),
  voice-messaging                         (33) -- reserved

  -- Content types beyond and including 64 are for
  -- bilaterally-agreed use between peers.
} (0..127);


-- If this message was originated as an RFC-822 message, then this
-- EMSDMessageId shall be the "Message-Id:" field from that message.
-- If this message was originated within the EMSD domain,
-- then this identifier shall be unique for the Message Center
-- generating this id.

EMSDMessageId ::= CHOICE
{
  emsdLocalMessageId     [APPLICATION 4]  IMPLICIT
                         EMSDLocalMessageId,
  rfc822MessageId        [APPLICATION 5]  IMPLICIT
                         AsciiPrintableString
                         (SIZE (0..ub-message-id-length))

};
```

```
EMSDLocalMessageId ::= SEQUENCE
{
  submissionTime                  DateTime,
  messageNumber                   INTEGER (0..ub-local-message-nu)
};

-- An Originator/Recipient Address in EMSD Environment

EMSDORAddress ::= CHOICE
{
  -- This is the local-format address
  emsd-local-address-format            EMSDAddress,


  -- This is a globally-unique RFC-822 Address
  rfc822DomainAddress                 AsciiPrintableString
};



EMSDAddress ::= SEQUENCE
{
  emsd-address         OCTET STRING
                                (SIZE (1..ub-emsd-address-length)),

  -- emsd-address is a decimal integer in BCD (Binary Encoded Decimal)
  -- format.
  -- If it had an odd number of digits, it is padded with 0 on
  -- the left.

  emsd-name                 [0]     IMPLICIT OCTET STRING
                                    (SIZE (0..ub-emsd-name-length))
                                    OPTIONAL
};

DateTime ::= INTEGER;

Iso8859String ::=  GeneralString;

AsciiPrintableString ::= [ APPLICATION 0 ]
                         IMPLICIT Iso8859String (FROM

    (" "|"!"|"#"|"$"|"%"|"&"|"'"|"("|")"|"*"|"+"|","|"-"|"."|"/"|
     "0"|"1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9"|":"|";"|"<"|"="|">"|
     "?"|"@"|"A"|"B"|"C"|"D"|"E"|"F"|"G"|"H"|"I"|"J"|"K"|"L"|"M"|
     "N"|"O"|"P"|"Q"|"R"|"S"|"T"|"U"|"V"|"W"|"X"|"Y"|"Z"|"["|"]"|
     "^"|"_"|"`"|"a"|"b"|"c"|"d"|"e"|"f"|"g"|"h"|"i"|"j"|"k"|"l"|
     "m"|"n"|"o"|"p"|"q"|"r"|"s"|"t"|"u"|"v"|"w"|"x"|"y"|"z"|"{"|
     "|"|"}"|"~"|"\"|""""));
```

```
ProtocolVersionNumber ::= [APPLICATION 1]    SEQUENCE
{
  version-major                   INTEGER,
  version-minor           [0]     IMPLICIT INTEGER DEFAULT 0
}
END  -- end of EMSD-SubmissionAndDeliveryProtocol
```

B EMSD-IPM ASN.1 MODULE

This section compiles in one place the complete ASN.1 Module for EMSD-IPM.

```
EMSD-InterpersonalMessaging1995 DEFINITIONS ::=

BEGIN

IMPORTS EMSDORAddress, EMSDMessageId, AsciiPrintableString
  FROM EMSD-SubmissionAndDeliveryProtocol;

ub-recipients  INTEGER ::= 256;
ub-reply-to INTEGER ::= 256;
ub-subject-field INTEGER ::= 128;
ub-header-extensions INTEGER ::= 64;
ub-emsd-name-length INTEGER ::= 64;
ub-mime-version-length INTEGER ::= 8;
ub-mime-content-type-length INTEGER ::= 127;
ub-mime-content-id-length INTEGER ::= 127;
ub-mime-content-description-length INTEGER ::= 127;
ub-mime-content-transfer-encoding INTEGER ::= 127;

IPM ::=   SEQUENCE
{
  heading                              Heading,
  body                                 Body OPTIONAL
};

Heading ::= SEQUENCE
{
  -- Address of the sending agent (person, program, machine) of
  -- this message. This field is mandatory if the sender
  -- is different than the originator.
  sender                      [0]     EMSDORAddress OPTIONAL,

  -- Address of the originator of the message
  -- (not necessarily the sender)
  originator                          EMSDORAddress,
```

```
-- List of recipients and flags associated with each.
recipient-data                      SEQUENCE SIZE (1..ub-recipients)
                                    OF PerRecipientFields,

-- Flags applying to this entire message
per-message-flags           [1]     IMPLICIT BIT STRING

{
   -- Priority values
   -- At most one of "non-urgent" and "urgent" may be specified
   -- concurrently.  If neither is specified, then a Priority
   -- level of "normal" is assumed.
   priority-non-urgent             (0),
   priority-urgent                 (1),

   -- Importance values
   -- At most one of "low" and "high" may be specified
   --  concurrently.  If neither is specified, then an
   -- Importance level of "normal" is  assumed.
   importance-low                  (2),
   importance-high                 (3),

   -- Indication of whether this message has been automatically
   -- forwarded
   auto-forwarded                  (4)
 }  OPTIONAL,

-- User-specified recipient who is to receive replies to this
-- message.
reply-to                    [2]     IMPLICIT SEQUENCE SIZE
                                    (1..ub-reply-to)
                                    OF EMSDORAddress OPTIONAL,

-- Identifier of a previous message, for which this message
-- is a reply
replied-to-IPM                       EMSDMessageId OPTIONAL,

-- Subject of the message.
subject                     [3]     IMPLICIT AsciiPrintableString
                                    (SIZE (0..ub-subject-field))
                                              OPTIONAL,

-- RFC-822 header fields not explicitly provided for in
-- this Heading. For messages incoming from the external
-- world (i.e. in RFC-822 format), the Message-Id: field
-- need not go here, as it is placed in the
-- Envelope's EMSDMessageId (message-id) field.
extensions                [4]   IMPLICIT  SEQUENCE
                          (SIZE (0..ub-header-extensions))
                                OF  IPMSExtension OPTIONAL,
```

```
  -- MIME Version (if other than 1.0)
  mime-version            [5]     IMPLICIT AsciiPrintableString
                                  (SIZE
                                  (0..ub-mime-version-length))
                                          OPTIONAL,

  -- Top-level MIME Content Type
  mime-content-type       [6]     IMPLICIT AsciiPrintableString
                                  (SIZE (0..
                                   ub-mime-content-type-length))
                                             OPTIONAL,

  -- MIME Content Id
  mime-content-id         [7]     IMPLICIT AsciiPrintableString
                                  (SIZE (0..
                                   ub-mime-content-id-length))
                                            OPTIONAL,

  -- MIME Content Description
  mime-content-description [8]    IMPLICIT AsciiPrintableString
                                  (SIZE (0..
                               ub-mime-content-description-length))
                                            OPTIONAL,

  -- Top-level MIME Content Type
  mime-content-transfer-encoding
                           [9]     IMPLICIT AsciiPrintableString
                     (SIZE (0..ub-mime-content-transfer-encoding))
                                               OPTIONAL
};

PerRecipientFields ::= SEQUENCE
{
  recipient-address                             EMSDORAddress,
  per-recipient-flags                           BIT STRING

   {
      -- Recipient Types.
      -- At most one of "copy" and "blind-copy" may be
      -- specified concurrently for a single recipient.  If
      -- neither is specified, than the recipient
      -- is assumed to be a "primary" recipient.
      recipient-type-copy                              (0),
      recipient-type-blind-copy                        (1),

      -- Notification Request Types.
      -- Only one of "rn" and "nrn" may be specified
      -- concurrently, \\x110011 for a single recipient.
      -- "rn" implies "nrn" in addition.
      notification-request-rn                          (2),
```

```
      notification-request-nrn                        (3),
      notification-request-ipm-return                 (4),

      -- Report Request Types
      -- At most one of these should be set for a
      -- particular recipient. "delivery" implies "non-delivery"
      -- in addition.
      report-request-non-delivery                     (5),
      report-request-delivery                         (6),

      -- Originator-to-Recipient request for a reply.
      reply-requested                                 (7)
   }  DEFAULT { report-request-non-delivery }

};

IPMSExtension ::= SEQUENCE
{
  x-header-label                      AsciiPrintableString,
  x-header-value                      AsciiPrintableString
};

Body ::= SEQUENCE
{
  compression-method          [0]     IMPLICIT CompressionMethod
                                               OPTIONAL,
  -- If compression method is not specified,
  -- "no-compression" is implied.

  message-body                        OCTET STRING
  -- See MIME for structure of the Body.
  -- If a compression method is specified, the entire text containing
  -- the Content-Type: element followed by the RFC-822 body are
  -- compressed using the specified method, and placed herein.
};

CompressionMethod ::= INTEGER
{
  -- Compression Methods numbered 0 to 63 are reserved for
  -- assignment within this and associated specifications.
  no-compression                  (0),
  lempel-ziv                      (1)

  -- Compression Methods numbered between 64 and 127 may be
  --  used on a bilaterally-agreed basis between peers.
} (0..127)

END  -- end of EMSD-InterpersonalMessaging1995
```

C RATIONALE FOR KEY DESIGN DECISIONS

This section summarizes the rationale behind key design decisions that were made while developing the EMSD Protocols.

C.1 Deviation From The SMTP Model

SMTP is the main mail transport mechanism throughout the Internet. SMTP is widely deployed and well understood by many engineers who specialize in Internet email. Because of these reasons, works based on SMTP or derived from it have a higher likelyhood of being widely deployed throughout the Internet.

However, SMTP is highly inefficient for transfer of short messages. SMTP's inefficiency applies to both the number of transmissions and also to the number of bytes transmitted.

Even when fully optimized with PIPELINING, SMTP is still quite inefficient.

Submission of a short message with SMTP involves 15 transmissions. Submission of a short message with SMTP and PIPELINING involves 9 transmissions. Submission of a short message with EMSD (EMSD-P and ESRO) involves 3 transmissions (in typical cases).

The key requirement driving the design of EMSD is efficiency. It was determined that the at least 3 fold gains in efficiency justifies the deviation from the SMTP model.

C.1.1 Comparison of SMTP and EMSD Efficiency

The table below illustrates the number of N-PDUs exchanged for transfer of a short Internet email when using SMTP, SMTP and PIPELINING, QMTP and EMSD. The names used for identifying the PDUs are informal names.

```
        SMTP      SMTP + pipelining   QMTP, QMQP,   EMSD
        -------   -----------------   ------------  -----------
client: SYN       SYN                 SYN           Submit.Req
server: SYN ok    SYN ok              SYN           Submit.Resp
client: HELO      EHLO                message       ack
server: ok        PIPELINING          accept close
client: MAIL      MAIL RCPT DATA      close
server: ok        ok
client: RCPT      message QUIT
server: ok        accept ok close
client: DATA      close
server: ok
client: message
server: accept
client: QUIT
server: ok close
```

```
client: close
```

C.2 Use of ESRO Instead of TCP

In order to provide the same level of reliability that the existing email protocols provide for short messages, it is clear that a reliable underlying service is needed. UDP [?], by itself, is clearly not adequate.

Use of TCP however, involves three phases:

1. Connection Establishment
2. Data Transfer
3. Disconnect

Reliable transfer of a short message using TCP at a minimum involves 5 transmissions as it is the case with QMTP.

The key requirement driving the design of EMSD is Efficiency. It was determined that elimination of the extra 2 transmissions that are an inherent characteristic of TCP, justifies deviation from it.

ESRO protocol, as specified in (RFC-2188 [?]), provides reliable connectionless remote operation services on top of UDP [?] with minimum overhead. ESRO protocol supports segmentation and reassembly, concatenation and separation.

Reliable transfer of a short message using ESRO involves 3 transmissions as it is the case with EMSD-P.

C.3 Use Of Remote Procedure Call (RPC) Model

Many Internet protocols are ”text-based”. Few Internet protocols are RPC based. Protocols designed around the ”text-based” approach have a better track record of acceptance throughout the Internet.

Considering that message submission and delivery in EMSD involve no more than two data exchanges, the text-based model becomes the same as an operation. Furthermore, the RPC model is the natural way of using ESRO.

C.4 Use Of ASN.1

In order to minimize the number of bytes transferred, efficient encoding mechanisms are needed.

Amongst today’s encoding mechanisms, ASN.1 has the unique feature of separating the abstract syntax from the encoding rules. By selecting ASN.1 as the notation used for expressing EMSD’s information objects, EMSD has the flexibility of using the most efficient encoding rules such as Packed Encoding Rules (PER) when they are available.

Efficient encoding can always be better performed when the syntax of the information is known. In general, encoding and compression techniques which use the knowledge of the syntax of the information produce better results than those compression techniques that work on arbitrary text.

D FURTHER DEVELOPMENT

Beyond this documentation of existing implementations, further development of EMSD protocol is anticipated.

The following deficiencies and areas of improvement are identified.

- Mapping of RFC-822 to EMSD-FS needs to be more explicit.
- Mapping of EMSD-FS to RFC-822 needs to be more explicit.
- Text of duplicate detection section needs more structure.
- SubmissionControl operation needs more informative description.
- Based on implementor's feedback the "EMSD PROCEDURE FOR OPERATIONS" section needs to be adjusted or re-done.
- The EMSD protocol can be extended to also support transfer of raw RFC-822 text-based messages in addition to EMSD-FS. This would be a trade-off in favor of "ease of implementation" against "efficiency of bytes transfered".
- Provide mechanisms to support fully automated initial provisioning of mail-boxes.

Future development of the EMSD Protocol is anticipated to take place at http://www.emsd.org/. Those interested in further development and maintenance of this protocol are invited to join the various mailing lists hosted at http://www.emsd.org/.