NEDA-EMSD (SA and UA)
Design and Implementation Notes

Platform: UNIX

Neda Document Number: 105-102-01

Last Updated: Author unspecified

Doc. Revision: source unspecified

Neda Communications, Inc.

August 1995

Contents

List of Figures

Preface

This document is a living document reflecting work in progress. It is intended to be used to document the programming development environment of the Neda implementation of a reference system employing the Efficient Mail Submission & Delivery (EMSD) protocols. It will document the design of the system and the code written to implement those protocols. A reference system will show only that the protocols do work as expected under the environment used with this specific implementation.

The scope of this document is the protocols of Efficient Mail Submission & Delivery (EMSD). It does not include the ESROS (Efficient Short Remote Operation Services) protocols.

The purpose of this document is to verify the design and implementation of EMSD Server Agent.

This document is a draft. It is nearly complete but not flawless. There are some sections that are not complete. They are yet to be written. They are marked as TBD (to be defined).

This document should, however, be fully correct in what it does say. It is therefore open to criticism on anything it does include — from specific examples and descriptive text, to the ordering of chapters and sections. If something is confusing or incorrect, then it should be fixed. Please let us know.

As you use the document, please mark pages with corrections so you can later look them up and send them in. Please refer any comments to the chapter name and section name, since page numbers and chapter and section numbers will change.

Distribution of this document in its present form should be limited to the parties directly involved in planning and deployment of EMSD.

Who to contact

```
For more information about this document or its contents, please
contact:

Neda Communications, Inc.
Phone: 1+(425) 644-8026
Fax: 1+(425) 562-9591
E-Mail: mohsen@neda.com
```

1 Introduction

This document explains the details of design and implementation of Neda Reference Implementation of Efficient Mail Submission and Delivery Protocol which conforms to the EMSD Protocol Specifications[**?**]. Documentation for ESROS (Efficient Short Remote Operation Services) is excluded from this document.

This document is prepared for designers and developers of messaging systems that conform to EMSD Protocol Specifications.

The main purpose of the reference implementation of EMSD is to verify the correctness and completeness of the EMSD protocol specifications. The reference implementation is also intended to facilitate and expedite the realization of the CDPD messaging system.

This reference implementation is intended to fulfill all network side requirements for the early stages of the evolution of the network. Complete sources for both the Network and the Device side are implemented. No specific accommodations will be made for the final form device realization. The reference implementation is developed under Solaris 2 and is targeted for the Solaris 2 environment.

This document will make little sense to anyone not extremely familiar with EMSD Protocol Specifications[**?**]. There has been no attempt in this document to re-explain anything previously covered in the Protocol Specifications[**?**]. Familiarity of Open C Platform[**?**] and ESROS[**?**] are also highly

2 Architecture and Design

Figure 1, Reference Implementation Architecture, depicts the architecture of the reference implementation. All software for the reference implementation is developed under Solaris 2.

The following legend applies to Figure 1

The amount of shading in each box roughly signifies the extent of development effort (white=none, light=little, dark=lots) for that module.

Under the reference implementation, a message would be created on the EMSD device using the PINE user agent. It would be written to disk, awaiting to be picked up by the polling EMSD-User Agent Daemon (EMSD-UA Daemon). Through the EMSD API, the EMSD-UA Daemon (User Daemon) uses the ES-OPERATION to accomplish the submission and delivery of the EMSD message to the EMSD Message Center. Using the EMSD Protocols, the ES-OPERATION passes the message to its peer process, ES-OPERATION on the system machine. Again using the EMSD API the message is exchanged with the EMSD-Server Agent Daemon (EMSD-SA Daemon). The EMSD-SA Daemon converts the message to RFC-822[**?**] format and submits it to sendmail.

When sendmail receives a message which is bound for an EMSD user, it delivers it using the EMSD-Mailer. The EMSD-Mailer knows to write it to disk so that it can be picked up by the EMSD-SA Daemon (System Daemon)and converted from RFC-822[**?**] format to the EMSD format. The message then simply reverses the same path as the EMSD-created message until it reaches the EMSD-UA Daemon. When the EMSD-UA Daemon receives a message it writes it to disk, to be picked up by PINE.

2.1 MAIL User Agent (PINE)

PINE is a publicly available user agent. It has been slightly modified to send messages to disk to be picked up by the EMSD-UA Daemon and to receive messages from disk written by the EMSD-UA Daemon.

2.2 EMSD-User Agent Daemon (EMSD-UA Daemon)

EMSD-UA Daemon will have two components: emsd-ua-deliver and emsd-ua-submit. The From address and other EMSD address translations are made here.

emsd-ua-submit converts the RFC-822[**?**] message into EMSD-Format Standard (EMSD-FS). Then, through the EMSD API uses the ES-OPERATION to accomplish submission and delivery of the EMSD message to its peer entity on the system side.

emsd-ua-deliver converts the EMSD-FS into RFC-822 and then writes it to disk to be made available for the PINE user agent.

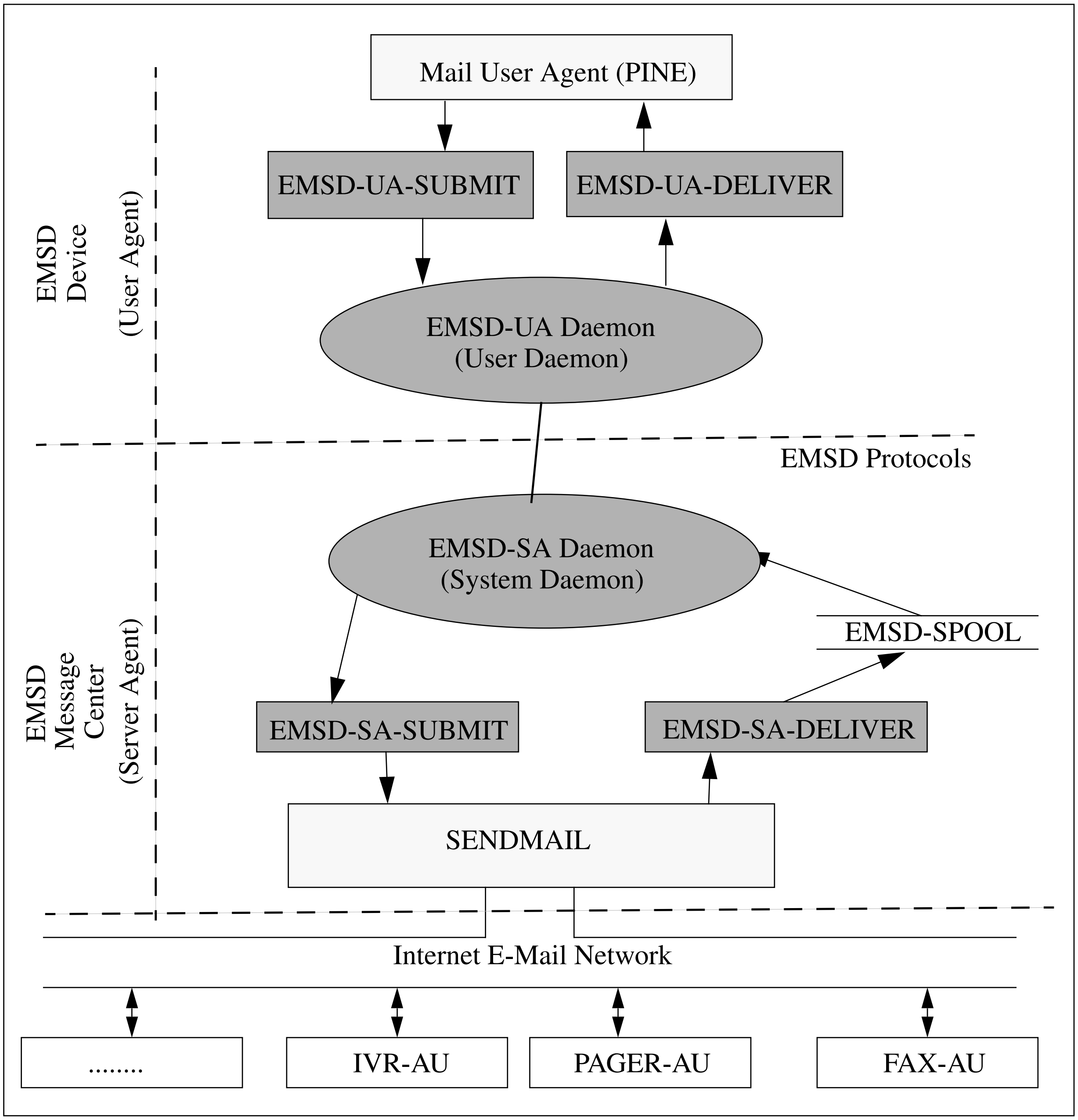

Figure 1: Reference Implementation Architecture

2.3 ES-OPERATION

ES-OPERATION is responsible for implementation of EMSD Protocol on both the device and system sides. ES-OPERATION exposes the EMSD API to its users, the EMSD-UA Daemon and the EMSD-SA Daemon.

2.4 EMSD-Server Agent Daemon (EMSD-SA Daemon)

As with the EMSD-UA Daemon, the EMSD-SA Daemon will have two components: emsd-sa-deliver and emsd-sa-submit. The From address and other EMSD address translations are made here.

emsd-sa-deliver converts the RFC-822 message into EMSD-FS. Then, through the EMSD API uses the ES-OPERATION to accomplish submission and delivery of the EMSD message to its peer entity on the user side.

emsd-sa-submit converts the EMSD-FS into RFC-822 and then passes it to sendmail.

2.5 EMSD-Mailer

The EMSD-Mailer is a mailer created to enable sendmail to deliver EMSD bound messages.

2.6 SENDMAIL

The standard sendmail has been slightly modified to add a new mailer. This new mailer is linked to ”emsd.neda.com”. The mailer runs ”emsd-sa-deliver”.

3 Split Device User Agnet

See Split Device document.

3.1 Split Device Message Submission

3.1.1 Mail User Agent - Compose and Send

Starts an SMTP session.

3.1.2 frontmail

Frontmail takes the message and puts it in qmail queue as normal message.

3.1.3 qmail-remote-emsd-ua-submit.sh

Replacement for qmail-remote.

Then call emsd-ua-submit.

3.1.4 emsd-ua-submit

Puts the message in the emsd-ua-daemon queue.

3.2 Split Device Message Delivery

3.2.1 emsd-ua-daemon

Gets the message with EMSD protocol.

3.2.2 emsd-ua-deliver

Does a qmail-inject destined to a local address.

3.2.3 qmail

qmail does the delivery to a maildir.

3.2.4 courier-imap

Courrier serves the message.

3.2.5 MUA – message retrival

4 EMSD-SA Qmail Integration

4.1 EMSD-SA Message Submission

Receives the message over the net emsd protocol.

4.1.1 emsd-sa-submit

Calls qmail-inject

4.1.2 qmail

qmail does the forwarding/processing.

4.2 EMSD-SA Message Delivery

4.2.1 Internet Mail Address

Arrives at emsd@user.example.net

In .qmail-emsd emsd-sa-deliver (emsd-sa-mailer) is called.

4.2.2 EMSD-SA-DELIVER

emsd-sa-mailer.

4.2.3 EMSD-SA-Daemon

Establishes the connection and does the delivery and retries.

5 qmail-Emsd Spooler Re-Design – NEW

5.1 qmail-Emsd-SA Spooler

sets up a separate /var/qmail-emsd/bin directory

Its own timers and itsown scripts.

qmail-remote replacement.

5.1.1 EMSD-SA Deliver Program – Not Daemon

5.1.2 EMSD-SA Submit Daemon

5.2 qmail-Emsd-UA Spooler

sets up a separate /var/qmail-emsd/bin directory

Its own timers and its own scripts.

qmail-remote replacement.

5.3 EMSD-UA Submit Program – Not Daemon

5.4 EMSD-UA Deliver Daemon

6 Submission and Delivery Module

Submission and delivery of messages in the EMSD environment requires cooperation between the EMSD-SA Daemon and the EMSD-UA Daemon. When messages are to be submitted, a device creates a message, and "hands" it to the EMSD-UA Daemon. The User Daemon creates an EMSD-formatted message which is passed to the EMSD-SA Daemon (via the services of ESROS). The System Daemon parses the EMSD-formatted message, creates a new RFC-822-format message, and passes the message to an SMTP entity for transfer to the recipients.

Conversely, when the local SMTP routing entity has a message to be delivered to a device/user, the SMTP entity calls an EMSD Mailer, which "hands" the message to the EMSD-SA Daemon. The System Daemon creates an EMSD-formatted message, which is passed to the EMSD-UA Daemon (via ES-OPERATION). The User Daemon parses the EMSD-formatted message, creates a new RFC-822-format message, and appends it to the user's mailbox.

The EMSD-SA Daemon has provisions for queuing messages, so that if a message can not be delivered immediately, a number of retries can be made at various intervals, to deliver the message. If, after a pre-defined number of retries have been made, the message has still not been delivered, a non-delivery message is created and sent back to the non-delivered message's originator.System Daemon Operation.

6.1 System Daemon Operation

The System Daemon sits idle most of the time. When an event occurs, it wakes up and handles the event. Events are of two types:

- It's time to scan for new messages
- It's time to process a queued message

At start-up time, a timer is created which will wake up the System Daemon periodically to look for new messages. When a message is to be delivered, it is placed in a directory known as the New Message directory. When the scan for new messages timer expires, the System Daemon wakes up, and scans the New Message directory to see if there are any new messages. When new messages are found, each message is moved from the New Message directory into the Queue Directory. At the same time, a process queued message timer is created and associated with this message. The timer is set to expire immediately, which will cause a new event to occur to wake up the System Daemon, at which time it will process the queued message.

When the timer expires, an attempt is made to deliver the message associated with the timer. If this delivery succeeds, the message is deleted from the Queue Directory. If the delivery fails, and the maximum number of retries has not yet been reached for this message, a new process queued message timer is created and associated with this message. The timer expiration time is set to the next retry time for this message. (Retry times are described below.)

If the System Daemon determines that the maximum number of retries has already been reached, a non-delivery message is created and queued for delivery to the non-deliverable message's originator.

The System Daemon handles two clients. One client is the EMSD Mailer which requests message delivery, on behalf of the SMTP world, to the EMSD world. The other client is a sub-process of the System Daemon which listens for EMSD indications and creates messages for submission, on behalf of the EMSD world, to the SMTP world. Each of these clients has its own New Message directory and its own Queue Directory.

6.2 User Daemon Operation

The User Daemon operates the same as the System Daemon, with the following exceptions:

- The first client of the User Daemon is the PINE user agent, vs. the EMSD Mailer for the System Daemon.
- Messages are passed by the PINE user agent to the User Daemon for Submission, vs. messages are passed by the EMSD Mailer to the System Daemon for Delivery.
- Non-delivery messages are not generated by the User Daemon.

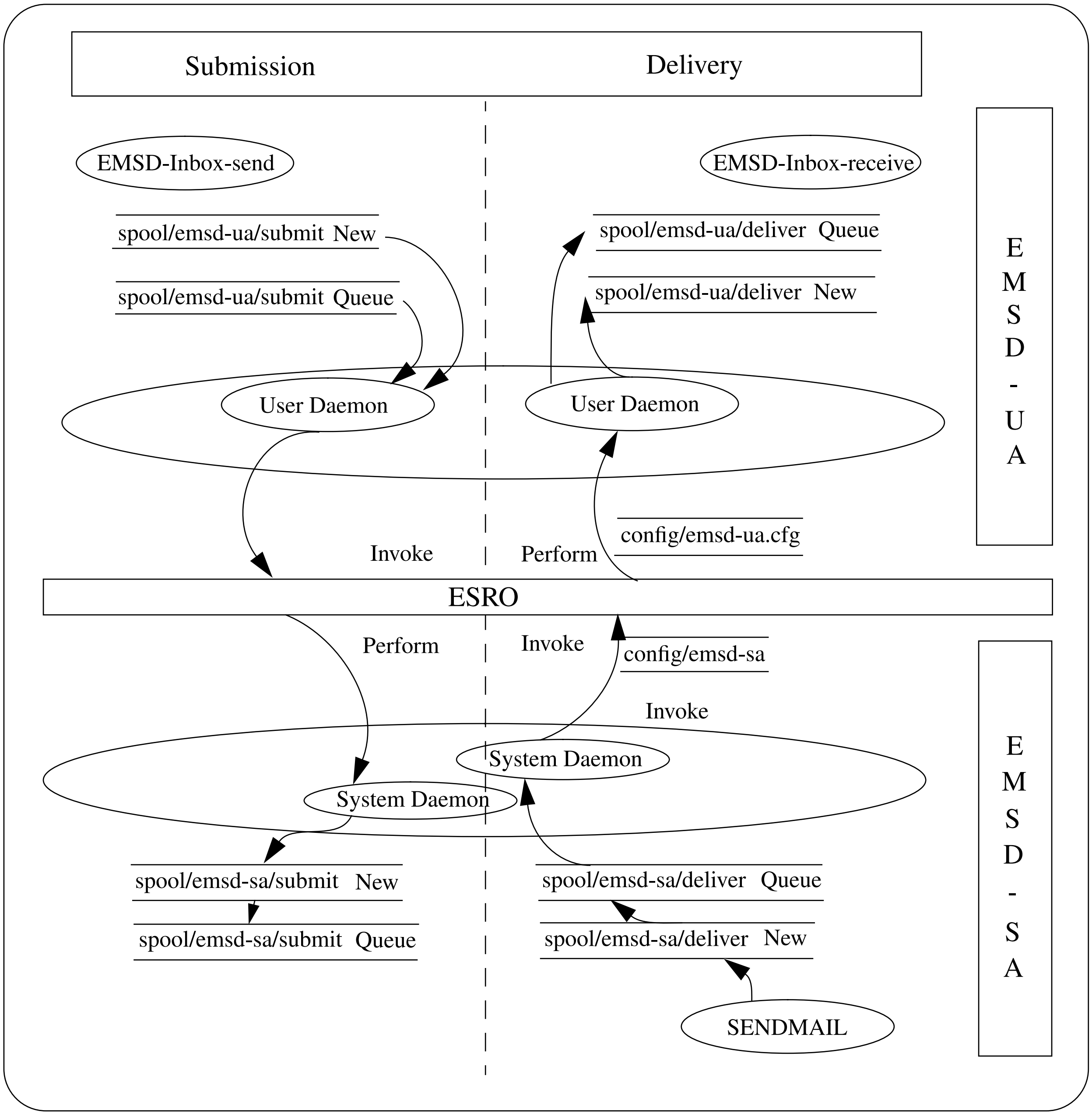

Figure 2: Submission and Delivery Module

6.3 EMSD Mailer Operation

When the Unix mailer (typically sendmail) has messages destined to the EMSD world, it is configured to call the EMSD Mailer. The recipient names, for which the EMSD world is responsible for delivery, are taken as command-line parameters by the EMSD Mailer. The complete RFC-822 message is passed to the EMSD Mailer as standard input. Upon receiving this information, the EMSD Mailer creates files, as described below, in the System Daemon's New Message directory.

6.4 File Formats

This section will describe the format of files used by the EMSD-SA Daemon (System Daemon) and the EMSD-UA Daemon (User Daemon). These files include:

- Configuration Files read at start-up time
- Data and Control files created for each sent message

6.4.1 Configuration Files

Each daemon application has a configuration file that it reads at start-up time, to determine the location of its two client's New Message directories and its Queue directories, and, in the case of the System Daemon, the local host name, and the host name of the machine to which an SMTP connection is to be established for messages outbound from the EMSD world to the SMTP world.

The configuration file for the User Daemon has a very simple format. It contains four lines, each with a single directory path. The four lines are as follows:

- Client 1 new message directory (from RFC-822)
- Client 1 queue directory
- Client 2 new message directory (from EMSD)
- Client 2 queue directory

The configuration file for the System Daemon contains the following additional entries:

- Local host name
- Host to which SMTP connection should be established

6.4.2 Data and Control Files

Two files are created for each message to be sent. The first is the message Data file. This file contains the contents of the message which is to be submitted or delivered. The second file is a Control file. The purpose of the Control file is to specify:

- The number of delivery attempts made so far on the message

- The number of previous retry attempts
- The time for the next retry attempt
- The name of the Data file
- The names of the recipients of the message for whom the daemon is responsible

Whenever a message is created, the following steps are taken:

1. The Data file, containing the complete message, is created in the appropriate directory. Data files are named beginning with "D_", followed by a number making the file name unique.
2. A temporary file is created in the same directory, and the Control file data is written to this temporary file. Temporary files are named beginning with "T_", followed by a number making the file name unique.
3. The temporary file is renamed to be a Control file. Control files are named beginning with "C_", followed by a number making the file name unique.

6.5 Configured Parameters

Configurable parameters are of three distinct types: Run-time, Compile-time, and Code Modification.

6.5.1 Run-time configurable parameters

The configuration files for the System Daemon and User Daemon contain the names of the directories which are to be used for new and queued messages.

6.5.2 Compile-time configurable parameters

The following parameters may be specified at compile time:

6.5.3 Number of Retries

This is the number of retries to deliver a message, prior to sending a non-delivery message to the originator.

The number of retries is determined by the number of entries in the array retryTimes in the file emsd/daemon/daemon.c

Individually, the amount of time between each retry may be specified. This allows for regressive back-off of retries, so that the first few retries may be attempted with little delay between them, while later retries wait much longer between delivery attempts.

Each entry in the array retryTimes in file emsd/daemon/daemon.c is the next amount of time to wait, before attempting to send a message the next time. The values in this array are in units of seconds, but the macros MINUTES, and HOURS, defined immediately prior to retryTimes, may be used to easily specify longer time periods.

6.5.4 New Message Directory

The New Message directory in which the EMSD Mailer places new messages is hard-coded (at present), and is therefore a compile-time configurable parameter.

To change the directory in which new messages are placed by the EMSD Mailer, modify the constant SPOOLDIR in the file emsd/apps/mailer.c

6.5.5 Code Modification

The User Daemon places all messages destined to a User in a file determined by the function getMailboxName() in the file emsd/apps/pebblesd.c. The current implementation always specifies the file name /tmp/emsd.mail. The parameters to the function getMailboxName() allow this function to provide a unique mail box file for each user. Additional code may be provided therein, to accomplish this.

6.6 Example Configuration Files

Example System Daemon Configuration File:

```
${EMSDRUN}/spool/emsd-mts/deliverNew
${EMSDRUN}/spool/emsd-mts/deliverQueue
${EMSDRUN}/spool/emsd-mts/submitNew
${EMSDRUN}/spool/emsd-mts/submitQueue
chaos.vis-av.com
amber.vis-av.com
```

Example User Daemon Configuration File:

```
${EMSDRUN}/spool/emsd-ua/submitNew
${EMSDRUN}/spool/emsd-ua/submitQueue
${EMSDRUN}/spool/emsd-ua/deliverNew
${EMSDRUN}/spool/emsd-ua/deliverQueue
```

6.7 Sample Message Flow

The following message flow occurs as a message is submitted by a user:

- User creates a message (using modified Pine).
- When the Pine user selects Send, Pine looks in the User Daemon's configuration file to determine the name of the RFC-822 New Message directory that it should create its files in. Using the configuration files shown above, Pine would select /usr/spool/emsd/emsdNew as the directory in which it should place its files.
- Two files are created in this directory. The control file specifies that this is the first attempt at sending this message, that there were no previous attempts to send the message, that it should be sent immediately, the name of the data file, and the names of the recipients to whom this message is being sent. The data file contains the RFC-822 encoding of the message to be submitted.

- Upon determining that there is a control file in its RFC-822 New Message directory, the User Daemon reads the control file to determine the name of the associated data file, moves the data file to the RFC-822 Queued Message directory, creates a new control file for this message (with the new full path name of the data file), and creates a timer which will expire immediately, in order to process this message.
- When the timer expires (immediately), the User Daemon obtains the control file name associated with the timer expiration, reads the data therein, and opens and parses the associated data file (which is encoded in RFC-822 format). Assuming that it is able to parse the message, the message is converted to EMSD format, and passed to ES-OPERATION, for transmission to the EMSD System Daemon. If ES-OPERATION was unable to transfer the message at this time (e.g. the EMSD System Daemon was not running), then the number of send attempts is incremented, a retry time is determined based on the number of previous send attempts, and the control file is recreated with this new information. If the message was transferred to the EMSD System Daemon, then both the control file and the data file are deleted.
- The EMSD System Daemon receives the message from the EMSD User Daemon via ES-OPERATION. When it receives the message, it parses it, to determine that it is in proper EMSD format, creates a data file in the EMSD New Message directory, to which it writes the message data, creates a control file indicating that this message has no prior send attempts and that it should be tried immediately, and creates a timer which will expire immediately, in order to process this message.
- When the timer expires (immediately), the System Daemon obtains the control file name associated with the timer expiration, reads the data therein, and opens and parses the associated data file (which is encoded in EMSD format). The message is converted to RFC-822 format. A connection establishment is attempted, with the specified SMTP entity through which the message should be routed. If a connection is established, and the message can be sent, then the control file and the data file are deleted. Otherwise, a new control file is created, with an updated retry count and next retry time.

A similar message flow exists for message delivery, in the opposite direction. The differences are as follows:

- When the system mail daemon (sendmail or alternative) has messages to be routed to the EMSD environment, it calls the EMSD Mailer with the message as standard input, and the EMSD Recipient Names as parameters. The EMSD Mailer creates a data file in the EMSD System Daemon's RFC-822 New Message directory. Additionally, it creates a control file.
- When the System Daemon notices a control file in its New Message directory, it reads the file, moves the data file to the Queued Message directory, creates a new control file, and creates a timer for immediate expiration.
- The remainder of the flow works similarly to, but in reverse of, the flow for submitted messages, until the User Daemon is to deliver the message to the user. In this case, instead of establishing a connection with an SMTP entity, the message is simply placed in a mail box file, the name of which is determined based upon the Recipient Name. Pine then retrieves messages for the user from that mail box file.

7 Interface to Standard MTAs

7.1 EMSD-Mailer

The EMSD-Mailer is a mailer created to enable sendmail to deliver EMSD bound messages.

7.2 SENDMAIL

The standard sendmail has been slightly modified to add a new mailer that is linked to ”emsd.neda.com”. The mailer runs ”emsd-sa-deliver”.

A Acronym

ASN.1	Abstract Syntax Notation One (ASN.1)
FSM	ESROS Finite State Machine.
IP-Message	InterPersonal Message
ESROS	Efficient Short Remote Operation Services
EROP	ESROS Protocol Engine
ESRO-SAP	ESROS Service Access Point.
MD	Management Domain
MH	Message Handling
MHS	Message Handling System
MS	Message Store
MT	Message Transfer
MTA	Message Transfer Agent
MTS	Message Transfer Service
SEQ_	Sequence Module
TMR_	Timer Management Module
TM_	Trace Module
DU_	Data Unit Management Module